AD-A037 226 HARRIS CORP MELBOURNE FLA ELECTRONIC SYSTEMS DIV F/G 9/4

MULTIPLEX SIMULATOR DESIGN STUDY.(U)

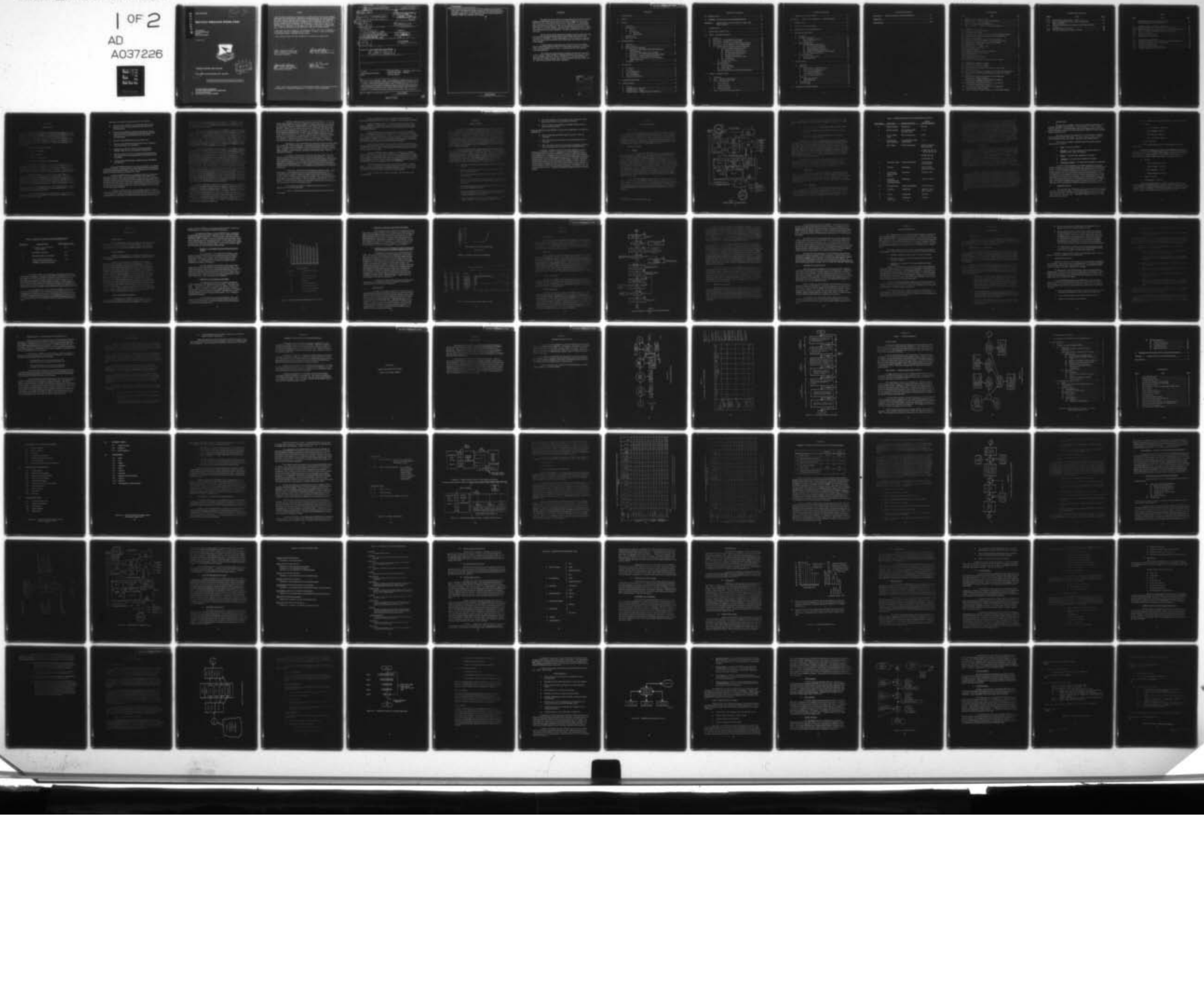
JAN 77 J A GRACIA, P J KNOKE F33615-73-C-1172

UNCLASSIFIED AFAL-TR-76-201 NL

1 OF 2

AD

A037226



12

AFAL-TR-76-201

MULTIPLEX SIMULATOR DESIGN STUDY

Harris Corporation
Electronic Systems Division
P.O. Box 37
Melbourne, Florida 32901

January 1977

TECHNICAL REPORT AFAL-TR-76-201

FINAL REPORT FOR THE PERIOD APRIL 1973 – JUNE 1976

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
Air Force Systems Command
Wright-Patterson Air Force Base, Ohio 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

James D Everett
JAMES D. EVERETT, Colonel, USAF
Chief, System Avionics Division
AF Avionics Laboratory

Pearl Hoskins
PEARL HOSKINS, Tech Mgr
Chief, Facility Engineering Group

James M. Riley
JAMES M. RILEY, Major, USAF
Chief, System Technology Branch
System Avionics Division

John E. Camp
JOHN E. CAMP
Electronic Engineer

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specified document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER 18 AFAL-TR-76-201	2. GOVT ACCESSION NO.
	3. RECIPIENT'S CATALOG NUMBER 9
4. TITLE (and Subtitle) 6 MULTIPLEX SIMULATOR DESIGN STUDY.	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report, Apr[redacted] 73 - Jun[redacted] 76.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 Jaime [redacted]. A. Gracia [redacted]. P. J. Knoke	8. CONTRACT OR GRANT NUMBER(s) 15 F33615-73-C-1172 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS Harris Corporation Electronic Systems Division P. O. Box 37, Melbourne, FL 32901	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 20030107
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Avionics Laboratory (AFAL/AAF) Air Force Wright Aeronautical Laboratories Air Force Systems Command Wright-Patterson AFB, Ohio 45433	12. REPORT DATE 11 Jan[redacted] 77
	13. NUMBER OF PAGES 194
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) 12 189p.	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

16 2003 17 01

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Simulator	Multiplex Techniques	Information Transfer System
Data Bus Resource Utilization	Simulator Host System	FORTRAN
GASP	Time Line Analysis	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report covers the definition, design, and development of MUXSIM (Multiplex Systems Simulator), a software package which performs computer-aided design and design verification of digital information transfer/multiplex systems. MUXSIM is primarily an aid for an organized approach at specifying and designing multiplex systems for diverse applications. It is a tool for obtaining answers to some complex digital multiplex system implementation problems, such as command and control techniques for data transfer, data bus requirements, → OVER

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408972

Cont

system interface hardware requirements, and data processing and manipulation requirements. The simulator is capable of adapting to new workload requirements and system configurations, as well as handling unknowns through employment of FORTRAN, GASP, and a simulator utility package.

FOREWORD

This report describes the results of a Study performed for the Air Force Avionics Laboratory by the Electronic Systems Division of the Harris Corporation under Contract F33615-73-C-1172, under Task 01, Work Unit 07, of Project 2003. Mr. Mark Thullen, Major James Riley, Dr. Michael O'Connor, and Mr. John Camp have served as Project Engineers for the Avionics Laboratory. The work was performed during the period 3 April 1973 through 11 June 1976, under the direction of Brian E. McIntosh and Jaime A. Gracia, Program Managers for the Harris Corporation. This report was submitted by the authors in June 1976.

The test case and related workload used during the simulator design phase were derived entirely from the DAIS Design Study (Contract F33615-74-C-1023 and Texas Instruments P.O. W776522). This was made possible by a fluid interleaving of study personnel who understood the requirements and therefore made a more effective contribution to both programs.

A related effort which was under way at the time of the report, IDAMST Software Specification (Contract F33615-76-C-1099, Boeing Company P.O. G-918746-9178), was using MUXSIM to derive results. It represents the start of the MUXSIM operational application.

The authors wish to acknowledge the significant contribution to the program of Messrs. Bryant P. Barnes, William M. Gulledge, William M. Hirt, Ronald M. Huhn, Brian E. McIntosh, and Irvin B. Slayton of Harris ESD, Dr. Donald Moon formerly of Harris ESD, Dr. Michael O'Connor, Capt. Fredrick Pensworth, Major James Riley, and Mr. Mark Thullen of the Air Force Avionics Laboratory.

CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

ILLUSTRATIONS

ILLUSTRATIONS (Continued)

TABLES

SECTION I

INTRODUCTION

The MUXSIM effort is primarily documented by this report, plus two other documents: the MUXSIM User's Manual (ref. 1) and the MUXSIM System Modification Design Data Manual (ref. 2) References 1 and 2 provide details of how to use and how to modify MUXSIM, respectively. Only summaries of this kind of information are included herein. The main body of this report is directed to present a broad overview of the entire effort, with emphasis on highlights. However, a considerable amount of detail on design specifics, history, rationale, etc. will be found in the appendices to this document.

The effort was divided into four phases:

- Phase I - Definition
- Phase II - Design
- Phase III - Implementation
- Phase IV - Operational Evaluation Database

The pertinent results and particulars of the first three phases are contained in the body of this Report and in Appendices A and B. The fourth phase was part of a larger distinct effort and was documented in a Technical Report, "Data Summary for IDAMST Baseline Avionic Suite," dated 17 February 1976 (ref. 9). Additional details may be found in the MUXSIM Technical Notes and Interim Reports cited in Appendix C.

MUXSIM was developed to fill the gap between manual analysis and breadboard techniques in the design of complex multiplex systems. It is basically a computer-aided design tool intended especially for simulation of multiplex systems. As such, it enables the multiplex system designer to reduce the time required to arrive at an optimal design, and enables him to debug his designs at an early stage to reduce the likelihood of costly hardware redesign efforts in the late stages of the development.

MUXSIM is not intended to replace all other existing tools of the multiplex system designer; rather, it is intended to supplement and complement them. Its main purpose may be stated succinctly as follows:

"To help in answering that set of multiplex system designer's 'what if' type questions which are difficult or impractical to answer accurately by other means and to do so in a cost-effective manner."

Among the critical MUXSIM design decisions were the following:

- Decision to focus MUXSIM on questions asked during the earlier phases of a typical multiplex system design effort (for maximum cost-effectiveness)

- Decision to focus MUXSIM on questions relating to the multiplex system architecture and organizational levels rather than on detail levels such as the electrical design of the bus (for maximum cost-effectiveness)

- Decision to make MUXSIM interactive (for ease of use)

- Decision to make MUXSIM Fortran-based (for ease of development, ease of modification, and "portability")

- Decision to use GASP as a vehicle for implementing dynamic MUXSIM models (for ease of development, and generality)

- Decision to emphasize the utility subsystem and the development of real workloads for driving MUXSIM models (for credibility of results and realism)

- Decision to make MUXSIM highly modular (for ease of development and flexibility)

The design of MUXSIM utilized mainly a top-down approach, with emphasis on modularity; however, in certain instances, a probe-coding (software breadboard) approach was used for sizing and feasibility studies. The rationale for all of the MUXSIM direction and tradeoff decisions is contained in this report.

Although the MUXSIM design and implementation effort has included many evaluations at different design levels, there has been limited time available for an operational evaluation of the completed package as a whole. However, several sample simulation experiments were performed as part of the MUXSIM verification effort which are thought to be typical of those which might be performed by a multiplex system designer. The results of these experiments, together with computer resources and user steps required for their execution, are covered in the report. They tend to indicate, but not to prove, the cost-effectiveness of MUXSIM.

In addition to MUXSIM's primary role as an analysis tool for use in early stages of the multiplex system design process, several MUXSIM outputs are directly useful in later stages also. In fact, some of these outputs can be used directly either as initial production aids or in facilitating modifications to production systems, such as might be required when changing or augmenting the avionics suite for an aircraft. For example, certain outputs aid in manufacturing operations, while others aid in multiplex system software development. Specific examples of MUXSIM benefits of different types are included in the body of the report.

From a functional or user's point of view, MUXSIM is an extensible software simulation package which is primarily intended as a tool for the multiplex system designer. It is presently implemented on the AFAL DEC System-10 under the TOPS-10 operating system. There, it runs interactively for maximum ease of use by the MUXSIM Basic User. However, because MUXSIM is a Fortran-based package, it is quite easily moved from one host system to another if the intended new host system has a Fortran compiler and a suitable disk operating system. A batch version of MUXSIM is also currently operational on the Harris Datacraft 6024/5 system. This version exists because it was convenient to develop MUXSIM by implementing and debugging it first on the Datacraft, followed by transfer to the AFAL DEC System-10 on a module-by-module basis.

From a logical or software implementer's point of view, MUXSIM consists of four major subsystems, termed the Utility, Static, Dynamic, and Executive, respectively. The Utility subsystem is essentially a data base management system for MUXSIM, where the data base consists of one or more signal flow lists which characterize the information transfer workload to be accomplished by the multiplex system being simulated. The Static subsystem consists of the RT assign work map, message map, fixed format scheduling and fixed format bus loading computation. It consists of eight models which represent eight different configurations of word/message mapping. This subsystem is called "static" because stochastic events are not explicitly considered, and because dynamic handling of simulated time is not required. Thus, the Static subsystem comprises mainly "steady-state" models of a multiplex system. The Dynamic subsystem consists of two models which are called "dynamic" because stochastic events characterizing such phenomena as multiplex system component failures, bus noise, and time-variable data transfer requirements are explicitly considered. This system is quite general, because it incorporates a complete discrete event and continuous simulation package called GASP IV as a component. Both Static and Dynamic subsystems are designed using a modular concept which allows new models to be easily added by the Advanced User of MUXSIM. The Executive subsystem provides the interface between the user and the other three subsystems; in addition, it provides coaching features for maximum ease of use in an interactive environment. The coaching features are omitted in the Harris batch version of MUXSIM.

From an operational or computer center manager's point of view, MUXSIM is a fairly large software package which requires substantial computer resources for storage and operation. Physically, it consists of about 11,700 Fortran statements (9,500 for MUXSIM's four subsystems and 2,200 for GASP). It is used in conjunction with 10,700 cards which describe the signal flow list for the A-7D aircraft. This signal flow list comprises the primary workload for the current version of MUXSIM; it describes about 2650 signals which characterize the information transfer requirements for a fairly complex aircraft. This workload was generated on DAIS Design Study. Another signal flow list for MUXSIM, called IDAMST (Integrated Digital Avionics for Medium STOL Transport), was developed by Harris as an extension of this contract. The IDAMST workload contains about 2200 signals and consists of 8200 cards.

MUXSIM is designed for ease of use but, like any other tool, it can be employed to best advantage by those with appropriate background and training. It is assumed that the MUXSIM Basic User has some knowledge of multiplex system design; such a person can realize useful results from MUXSIM with very little training in the operation of MUXSIM itself. However, for more sophisticated exercises, the Advanced User requires some specific training in MUXSIM and its components. For example, to create new static models he needs some knowledge of Fortran; and to create new dynamic models, he must familiarize himself with details of GASP. In short, although MUXSIM is designed to be easy to use, it is not fully automatic - it requires a good human driver. Furthermore, although MUXSIM may be modified or transferred to another host system with relative ease (i.e., it is "portable"), these operations also require some special knowledge. References 1 and 2 contain the information needed for any advanced use or modification of MUXSIM.

Regarding the future, MUXSIM has been enhanced by addition of another signal flow list (IDAMST, mentioned previously). Harris believes that operational usage will provide the best possible source of feedback on MUXSIM enhancement or modification requirements. In addition, the Harris MUXSIM implementation and verification efforts to date have already revealed a number of desirable improvement or enhancement areas. These are identified in the report.

In summary, this report gives an overview of the What, When, Why, and How of a multiplex system development and analysis tool called MUXSIM. In view of the continuing importance of multiplex systems in aircraft avionics systems design, there are many present opportunities for such use. Significant improvements and enhancements to MUXSIM now appear to be possible, but Harris believes that their specifics are best determined by feedback from actual operational use. In other words, Harris believes that future MUXSIM changes should be mostly evolutionary; it is so designed as to accommodate such changes with minimal cost.

In conclusion, MUXSIM has been developed and implemented after a careful design study by a team thoroughly familiar with the multiplex system design process. It is a simulation tool intended to enhance the capabilities of the multiplex system designer; it is not, however, intended to replace either him or his other tools. MUXSIM works, it is easy to use, and it is not excessively expensive to operate. Harris believes that MUXSIM is now ready for operational evaluation. Such an evaluation should show that it is cost-effective, and that it will lead to substantial development cost savings and reduced development cycle times for future aircraft multiplex systems.

The remainder of this report is organized into seven major sections, the contents of which are summarized below.

Section II (MUXSIM Purpose) deals with the motivation behind development of the system.

Section III (MUXSIM Description) is concerned with what MUXSIM is. Descriptions are provided from the functional, the logical, and the operational viewpoints.

Section IV (MUXSIM Benefits) is an attempt to answer the question "How valuable is MUXSIM?" In addition to a general value assessment, the section contains results of four specific simulation experiments which were conducted during the course of MUXSIM verification.

Section V (MUXSIM Use) discusses the general circumstances of MUXSIM use; i.e., the background requirements placed on the user, how the interactive features simplify use, typical use techniques, etc. Also covered are typical use costs, as reflected by the computer resource requirements for the conduct of the four simulation experiments mentioned above. MUXSIM use is not covered in great detail, because a comprehensive MUXSIM User's Manual exists which is solely devoted to that purpose.

Section VI (MUXSIM Modification) is concerned with how MUXSIM may be modified and/or moved from one host system to another. Again, this subject is not treated in great detail because a separate MUXSIM System Modification Design Data Manual has been created for that purpose.

Section VII (MUXSIM History) summarizes the development of MUXSIM from the initial concept, through design and development, to implementation and verification. Section VII addresses the question, "Where did MUXSIM come from?". Supplementary detail for the three phases of the effort is provided in Appendixes A and B, while an index to all technical notes, interim reports, etc. developed during the course of the MUXSIM effort is provided in Appendix C and in the Bibliography.

Section VIII (MUXSIM Future) deals with the question "Where is MUXSIM going?" It covers possible extensions to and improvements of MUXSIM, together with recommendations on methodologies for MUXSIM enhancement, as well as identification of some specific areas where further development now seems desirable.

Finally, Section IX is a summary of the report, plus its conclusions and recommendations.

SECTION II

MUXSIM PURPOSE

All simulators used as design tools basically answer "what if" types of questions, which are posed either indirectly or directly. For example, a simulation run typically consititutes part or all of a simulation experiment which is performed either as part of a trade-off study or as part of a design verification exercise. In the former case, the designer may wish to know which of several multiplex system design schemes leads to the best performance for a given information transfer workload. Here his question is: "What is P (performance) for a specified X (design alternative), given W (workload plus other aspects of the working environment)?" In the latter case his question is similar, but he may wish to assure himself that the design performs adequately over a range of workloads. Then the question becomes: "What is P (performance) for a specified W (workload), given X (a design alternative)?

The results of several specific simulation experiments of the type that can easily be run using MUXSIM are given in Section IV of this report, therefore such details will not be discussed here. Presently, the point to be made is that the initial version of MUXSIM was deliberately restricted in scope so as to easily and efficiently handle certain predefined classes of a multiplex system designer's questions, but not all possible imaginable questions that he might ask. This preliminary scoping was a primary output of the MUXSIM definition study, and is the key to MUXSIM's present cost-effectiveness. Examples of the questions to which MUXSIM is directly applicable are:

- How many remote terminals are required for a given information transfer workload?
- What is the optimum bus protocol for use in a typical fighter aircraft interior multiplex system?
- What is the bus loading associated with the particular bus command and control scheme?
- What is the minimum safe data bus speed for a given information transfer workload and bus command and control discipline, or how fast must the data bus be?
- What is the impact on bus loading resulting from the addition of several new avionics subsystems to the bus?
- What are the quantitative advantages of a particular multiplex system redundancy management scheme, given a certain EMI bus noise and multiplex subsystem failure environment?
- What is the impact on bus controller performance requirements of increasing information transfer workload for a fixed bus speed?

- What is the probability of lost or erroneous data for a particular redundancy design and given noise and failure environments?

- What is the impact on bus loading of a doubling of sample rates for a certain set of bus quantities?

Among the questions for which MUXSIM is usually either not applicable or only indirectly applicable are:

- What is the maximum permissible length of the bus for a given bus technology?

- What is the bit error rate for a given bus system design and noise environment?

- What is the point at which, from a life cycle cost standpoint, the multiplex system becomes more cost-effective than a hardwired system?

To summarize, MUXSIM is designed to be directly applicable to a set of multiplex system designer's "what if"? questions that cannot be readily answered by other available means. The version of MUXSIM that has been implemented is now directly applicable to a considerable number of such questions; and because of MUXSIM's modular architecture, it is readily extensible to cover a still broader domain of questions. The scoping issue is the key to MUXSIM's cost-effectiveness. In contrast, the lack of suitable scoping has been the key to failure of many simulators in the past. Because it is often possible to simulate systems at any desired level of detail, it is not unusual for a simulator designer to attempt to simulate "the universe"; and if he attempts to do so, the resulting simulator is likely to be very expensive to construct, very difficult to use, and excessively expensive to run. MUXSIM was deliberately designed to avoid this common problem.

SECTION III

MUXSIM DESCRIPTION

1. INTRODUCTION

In this section, summary descriptions of MUXSIM are given from three points of view: namely, the functional, the logical, and the operational. These viewpoints are intended to reflect the interests of the multiplex system designer, the MUXSIM implementer, and the MUXSIM operator, respectively. The descriptions are only summaries because, as was noted in the Introduction, details of MUXSIM descriptions from various viewpoints are provided in the MUXSIM User's Manual, in the MUXSIM System Modification Design Data Manual, and in the Appendixes to this report.

2. FUNCTIONAL VIEW

a. General

The functional view is MUXSIM as seen by its intended primary user, who is a multiplex system designer. He is interested in such questions as "When should I use it?" and "How do I use it?"; and he sees it mainly as an analysis tool. Specifically, it is a tool called a "discrete event software simulator"; and like most simulators intended for use in design applications, it helps to provide answers to a designer's "what if" type questions. Through its use, he is able to get faster and more accurate answers to a certain set of design questions than would have been possible without it. As has been mentioned earlier, MUXSIM is not intended to replace all of the designer's tools; rather, it replaces some and complements others. Although MUXSIM fills the gap between manual analysis and breadboarding techniques, it totally replaces neither. It was not intended to be, and is not, a universal analysis tool. However, because of its modular design and for other reasons, MUXSIM is highly flexible; it can therefore be easily modified and/or enhanced to perform analysis tasks which are not directly performable with MUXSIM as is presently implemented.

A top-level view of MUXSIM as seen by its user is given in Figure 1. This figure is a revised output of the MUXSIM definition phase*, and shows more than was actually scoped for implementation. However, that which was implemented is compatible with this figure, while that which was not implemented continues to be (potentially) desirable; therefore the figure can be regarded as both a conceptual level description of that which MUXSIM now is, and a blueprint for that which MUXSIM might become in the future.

*See Reference 5 and Appendix A of this report

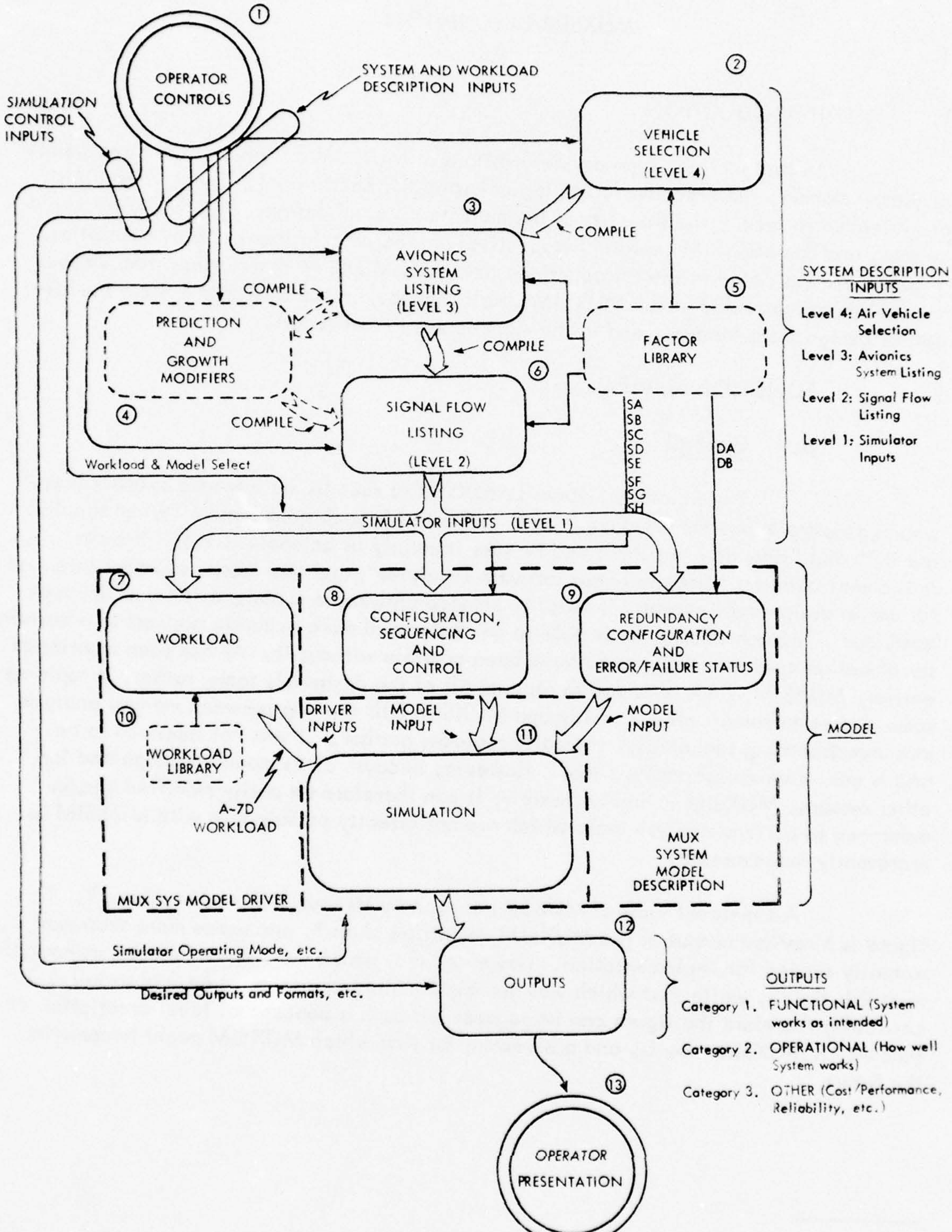

1
OPERATOR CONTROLS
SIMULATION CONTROL INPUTS
SYSTEM AND WORKLOAD DESCRIPTION INPUTS
2
VEHICLE SELECTION (LEVEL 4)
3
COMPILE
AVIONICS SYSTEM LISTING (LEVEL 3)
COMPILE
PREDICTION AND GROWTH MODIFIERS
4
COMPILE
COMPILE
5
FACTOR LIBRARY
6
SIGNAL FLOW LISTING (LEVEL 2)
SA
SB
SC
SD
SE
SF
SG
SH
DA
DB
Workload & Model Select
SIMULATOR INPUTS (LEVEL 1)
SYSTEM DESCRIPTION INPUTS
Level 4: Air Vehicle Selection
Level 3: Avionics System Listing
Level 2: Signal Flow Listing
Level 1: Simulator Inputs
7
WORKLOAD
8
CONFIGURATION, SEQUENCING AND CONTROL
9
REDUNDANCY CONFIGURATION AND ERROR/FAILURE STATUS
10
WORKLOAD LIBRARY
A-7D WORKLOAD
DRIVER INPUTS
MODEL INPUT
MODEL INPUT
11
SIMULATION
MUX SYSTEM MODEL DESCRIPTION
MUX SYS MODEL DRIVER
MODEL
Simulator Operating Mode, etc.
12
Desired Outputs and Formats, etc.
OUTPUTS
OUTPUTS
Category 1. FUNCTIONAL (System works as intended)
Category 2. OPERATIONAL (How well System works)
Category 3. OTHER (Cost/Performance, Reliability, etc.)
13
OPERATOR PRESENTATION

Figure 1

MUXSIM System - Conceptual Level

Briefly, a system designer wishing to use simulation as an analysis tool must somehow provide inputs which describe at least three types of simulation in part. These are:

- Inputs which describe the system being simulated, or system models;
- Inputs which describe the inputs to the simulated system, or model workloads;
- Inputs which describe the desired selection and form simulation outputs and simulator run modes.

The ease of use of a simulator can be measured by how easily the designer may specify such things as those listed. In Figure 2, these three types are called "system description inputs", "workload description inputs", and "simulation control inputs", respectively. They may be specified at many possible levels; four particular levels are shown in the figure. However, for the present version of MUXSIM, inputs are specified at Level 1. The other input levels, although usable, generally require significant manual effort. Also, there are three defined categories of simulator outputs shown in the figure, called "functional", "operational", and "other". For the present version of MUXSIM, most outputs are in categories 1 and 2, consistent with the simulator focus on the "systems" or "big picture" class of designer questions.

The interested reader will find a full description of Figure 1 in Appendix A, therefore such detail need not be repeated here. The present implementation and status for the 13 numbered functional boxes of Figure 1 is as summarized in Table 1. There it may be seen that of the 13 blocks identified, 11 have been either partially or completely built, while two were left for possible future implementation.

b. System Users

At this point, it is convenient to define two types of MUXSIM users, namely, the Basic User and the Advanced User. MUXSIM is designed to be easy for the former to use, but he is limited in what he can do with it. MUXSIM can be put to a much broader range of uses by the Advanced User, but he must be quite familiar with details of MUXSIM structure and other software items. We next consider these two types separately, Basic User first.

(1) Basic User

The Basic User normally selects a multiplex system model from a library of prepared models. (See Figure 1, Block 5, Factor Library.) He also selects a desired workload from the workload library and the outputs he wishes to see, and then sets the simulator into execution. In doing the above, he is substantially aided by the Executive subsystem which controls the interactions of the other subsystems, and which

Table 1. MUXSIM CONCEPTUAL HOST IMPLEMENTATION SYSTEM

Block Number	Block Name	Implementation Status	Relevant MUXSIM Subsystem
1	Operator Controls	Implemented	Executive
2	Vehicle Selection	Not Yet Implemented, Need More Data	-------
3	Avionics System Listing	Partially Implemented	Utility
4	Prediction and Growth Modifiers	Not Implemented, Need Historical Data	-------
5	Factor Library	Partially Implemented	Executive Subsystem, Static Subsystems ● Models SA, SB, SC, SD, SE, SF, SG, SH Dynamic Subsystem ● Models DA, DB
6	Signal Flow Listing	Partially Implemented	Utility Subsystem, A-7D Data Base
7	Workload	Implemented	Utility Subsystem, Executive Subsystem
8	Configuration, Sequencing and Control	Implemented	Executive, Static
9	Redundancy Configuration and Error/Failure Status	Implemented	Executive, Dynamic
10	Workload Library	Partially Implemented	Utility, Executive
11	Simulation	Implemented	Executive, Utility, Static, Dynamic
12	Outputs	Implemented	Executive
13	Operator Presentation	Implemented	Executive

also has a "coaching" mode which tells the basic user what models are available, what inputs need to be supplied, etc. In addition, the Basic User may modify an existing workload or define a new one, details concerning the accomplishment of which will be found in the MUXSIM User's Manual (which is primarily intended for the Basic User). Presently available to the Basic User are eight "static" models, two "dynamic" models, and one "workload". In Figure 1, the "static" models are shown as SA, SB ..., SH; the "dynamic" are DA and DB, and "workload" is the signal flow list for the A-7D aircraft. The adjectives "static" and "dynamic" were selected as terminology for models because the former represent a class of models for which the dynamic management of simulated time is not required (i.e., items that are "steady state"), while the latter represent a class in which dynamic management of simulated time is required. In general, dynamic models deal with stochastic events which occur at specific but unpredictable times, and which require specific system behavior in consequence of their occurrence.

(2) Advanced User

The Advanced User of MUXSIM has available to him all of the facilities available to the Basic User; in addition, he can modify existing models, add new models, and otherwise modify MUXSIM as he sees fit. Because MUXSIM is of modularized design and coded in Fortran and GASP, these additional tasks are not very difficult to perform. However, more detailed knowledge of MUXSIM implementation is required of the Advanced User if he wishes to perform these tasks. This information is available in the previously referenced MUXSIM System Modification Design Data Manual. In addition to providing information to the Advanced User of MUXSIM, the referenced manual, also provides information needed to make substantial MUXSIM enchancements, or to move MUXSIM from its present host system (the DEC System-10) to another host system. Because such a transfer is not difficult to perform, MUXSIM is said to be "portable."

In summary, the functional view of MUXSIM consists of MUXSIM as seen by its primary user, who is assumed to be a multiplex system designer. Such a person will typically use MUXSIM to answer "big picture" or systems-level types of questions. The answers he seeks are normally whether or not the system being simulated works at all, and if it does work, how well. The Basic User of MUXSIM uses it in unmodified form, except that he may change workloads; his primary reference manual is the MUXSIM User's Manual. The Advanced User of MUXSIM may modify the system, alter old models, add new ones, change outputs, etc., as he sees fit. His primary reference manual is the MUXSIM System Modification Design Data Manual. He may also require Fortran, GASP, and/or TOPS-10 reference manuals, depending on the particular changes or enhancements he wishes to make.

3. LOGICAL VIEW

The logical view of MUXSIM is that seen by the MUXSIM implementer. He is typically interested in such questions as - "How do I build it?", "How is it built?", "How do I verify it?". Such an implementer sees mainly software structure; and his view of this structure may be at high or low levels depending on whether he is mainly a software architect or mainly a coder.

The primary document which provides this view of MUXSIM is the MUXSIM System Modification Design Data Manual. This manual contains, in addition to flow charts and program listings, the functional specifications that MUXSIM realizes.

From this logical viewpoint, MUXSIM consists of a hierarchy of named software entities, as below:

- System - the whole of MUXSIM
- Subsystem - 1 of 4 major components of the system; namely, the Executive, Utility, Static, and Dynamic
- Program - 1 of several major components of a subsystem
- Subprogram - 1 of several major components of a program
- Subroutine, or Module - 1 of several major components of a subprogram

In addition to these components, the MUXSIM implementer sees internal MUXSIM interfaces as well as external interfaces (e.g., interface between MUXSIM and its users, and interfaces between MUXSIM and the operating system of its host computer system). To simplify MUXSIM development and verification, and to increase flexibility, modules have been kept small, with interfaces that are as clean as possible. Also, to keep MUXSIM as small as possible (in terms of number of lines of source code), modules are generalized and shared as much as possible. To make MUXSIM easy to use, considerable effort has been spent on the user interface; and to reduce MUXSIM development costs, considerable use has been made of the DEC System-10 operating system (TOPS-10). Some of the MUXSIM components are reentrant, while others are not; this is because there was no initial requirement that MUXSIM be usable by more than one user at a time.

4. OPERATIONAL VIEW

The operational view of MUXSIM is that view seen by the MUXSIM operator or by its host computer center manager. His typical questions are: "How do I get MUXSIM running on my system?", "How much of the computer resources are tied up by it? (disk space, core space, CPU time, etc.)", "How should I charge for its use?", and "How do I store it on the system?".

Physically, MUXSIM is totally described by about 11,700 cards, as follows:

- MUXSIM
  - Executive Subsystem, 1840 cards
  - Utility Subsystem, 2745 cards
  - Static Subsystem, 3660 cards
  - Dynamic Subsystem, 1245 cards
- GASP, 2200 cards

The A-7D data base which was used in the development of MUXSIM is 10,700 cards.

For reasons of economy, it may be best to store MUXSIM on magnetic tape. In this form, a 1200-foot reel of 800-bpi magnetic tape is sufficient for the job; if stored in punched card form, about six boxes of cards are needed. After compilation and link on the DEC System-10, MUXSIM consists of about 510 blocks of object code which will normally be kept resident on the DEC System-10 disk. The disk requirements are:

- MUXSIM
  - Executive Subsystem, 66 blocks
  - Utility Subsystem, 143 blocks
  - Static Subsystem, 177 blocks
  - Dynamic Subsystem, 124 blocks
- GASP, 133 blocks
- A-7D Data Base, 1699 blocks

When operated on DEC System-10, MUXSIM is normally segmented, and requires a partition of 30k 36-bit words for execution. CPU time requirements for typical use of the MUXSIM are difficult to estimate because a typical use of MUXSIM is difficult to define; however, as a guideline, the CPU time requirements for typical simulation experiments (discussed subsequently in Section IV) are shown in Table II.

Table II. SIMULATION EXPERIMENT RESOURCE REQUIREMENTS

Experiment #	Experiment Name	CPU Time Required (Min)
1	Bus Loading vs. Bus Command and Control Schemes	20.7
2	Bus Loading vs. Bus Speed	10.4
3	Controller Loading vs. Bus Loading	10.4
4	Impact of Command and Control Uncertainties on the Periodicity of the Fundamental Update Interval Starts	6.6

From the data in Table II, the typical costs of a simulation experiment may be estimated. For example, if we assume a cost of $8/min for DEC System-10 time, then the computer costs of the four experiments in question are about $165, $83, $83, and $53, respectively. These cost estimates are made for illustrative purposes only, and show that typical MUXSIM run costs are probably not excessive. Actual costs depend on the particular computer system accounting practices employed for the MUXSIM host system. They canof course vary widely. More details of interest in an operational view of MUXSIM may be found in the MUXSIM System Modification Design Data Manual (ref. 2).

As was mentioned earlier, a batch version of MUXSIM now exists which runs on the Harris Datacraft 6024/5. This is a moderate-size minicomputer system which includes 32k 24-bit words of core and a disk. The existence of this batch version shows that MUXSIM can be tailored to run on a fairly small computer system; however, since this version of MUXSIM is simply a byproduct of the MUXSIM development process and not a contractual end item, it is not presently described in available MUXSIM documentation.

SECTION IV

MUXSIM BENEFITS

1. INTRODUCTION

The main purpose of this section is to attempt to answer the question, "Of what value is MUXSIM?", both briefly and substantively. This is done by first summarizing the general benefits, and then describing four specific simulation experiments that have been run using MUXSIM, together with their results.

2. GENERAL BENEFITS

MUXSIM is mainly intended as a multiplex system designer's analysis tool. Specific examples of how MUXSIM can be helpful are provided in the following paragraphs.

In addition to its analysis role, MUXSIM can also be helpful in production and software stages of a multiplex system development. This is because several of its outputs are in such a form as to be directly usable as the multiplex system design moves toward these later stages. For the most part these outputs are helpful because without MUXSIM they would have to be produced by tedious and error-prone manual means, whereas MUXSIM generates them automatically. The remote terminal wiring list is an example of one such output; multiplex system bit, word, and message maps are other examples. In the former case, it is a necessary step in the production process to assure that each signal which is an output from one terminal is an input to another, and vice versa. While this is a conceptually simple signal accounting task, it is very tedious to perform manually. In the latter case, where the multiplex system is of the time-division-multiplex (TDM) type, it is necessary to know which signals are assigned to each field and which words are in each message in order to write or modify multiplex system software. This, too, is a conceptually simple task, but tedious to perform manually. (Note: In the A-7D signal flow list, there are 2650 signals, and this a moderate-size signal flow list as aircraft multiplex systems go. Some large aircraft have a signal flow list of 13,000 signals.)

3. SAMPLE SIMULATION EXPERIMENTS

For present purposes, a MUXSIM simulation experiment means "a planned sequence of MUXSIM runs intended to provide answers to one or more significant questions posed by a multiplex system designer." The ease and accuracy with which MUXSIM can be applied in obtaining answers to typical significant questions is a true measure of MUXSIM's effectiveness.

As contractually defined, the overall MUXSIM design, development and verification effort did not provide funding or time for a significant MUXSIM operational effectiveness evaluation. (This work is under way as part of contract F33615-76-C-1099.) However, as part of the MUXSIM verification efforts, several sample simulation experiments were performed which illustrate typical results which can be easily attained by the MUXSIM Basic User. Four such experiments, together with their results, are briefly summarized below. Three of them use static models, while the fourth uses one of the dynamic models provided.

a. Experiment 1 - Bus Loading Versus Bus Command and Control Schemes.

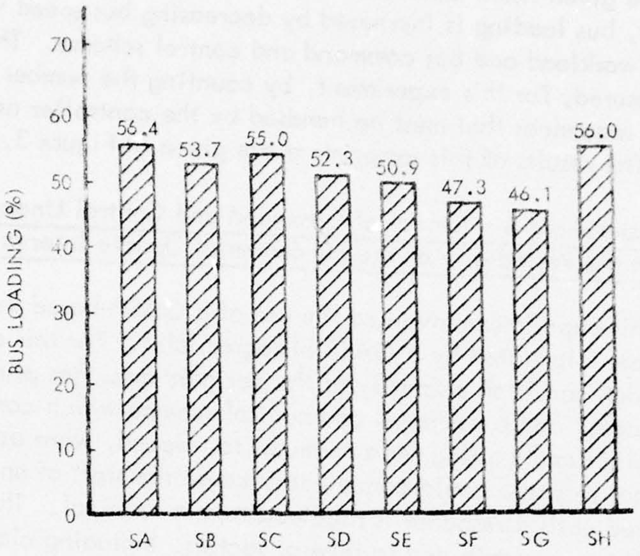
In this experiment, it was assumed that the multiplex system designer wishes to know which is the best bus command and control scheme for a given bus workload. He also wishes to know how much better one scheme is than another. For example, if a simple scheme is nearly as good as a more complex one, it may be preferable to use the simple one because it may be considerably less expensive to implement.

The present MUXSIM implementation consists of eight static models which represent eight different bus command and control schemes. This implementation covers a wide variety of configurations applicable to TDM systems, ranging from completely centralized (terminal-to-central-to-terminal) to completely distributed (direct terminal-to-terminal), and includes a number of hybrid combinations of both. Figure 2 indicates the results of an experiment run using the A-7D data base, which is being used to verify MUXSIM.

b. Experiment 2 - Bus Loading Versus Bus Speed.

This experiment assumed that the multiplex system designer wishes to explore details of suspected bus saturation effects. These effects take place in the vicinity of, or close to, 100 percent bus loading. The presence of saturation implies that for practical purposes a certain percent of the apparent bus capacities are unusable for the given bus scheduling algorithm. To investigate these saturation effects, bus speed is systematically varied while the bus workload and command and control schemes are held constant. (This is analogous to expanding the workload.)

The final results for the saturation experiment, which used a binary matrix scheduler, show that several fundamental update intervals within the given major frame become saturated at 92.9 percent overall bus loading for the particular implementation tested.



STATIC MUX MODELS

Model Name	Information Transfer Discipline
SA	T/T Transfer
SB	T/C/T Transfer (bit shuffling)
SC	Digital T/T, Discrete T/C/T
SD	Hybrid Transfer
SE	T/T Transfer with BCIU Broadcast
SF	T/C/T Transfer with BCIU Broadcast
SG	Hybrid Transfer with BCIU Broadcast
SH	T/C/T Transfer (word shuffling)

Figure 2. Bus Loading Versus Bus Command and Control Schemes

c. Experiment 3 - Controller Loading Versus Bus Loading.

In this experiment it was assumed that the multiplex system designer wishes to explore the impact of high bus loading on controller loading. He suspects that as bus loading increases, the controller may have to work harder in order to implement the given fixed command and control algorithm. As in the preceding experiment, bus loading is increased by decreasing bus speed while maintaining constant workload and bus command and control scheme. The controller loading is measured, for this experiment, by counting the number of different message group sequences that must be handled by the controller as the bus load is varied. The results of this experiment are given in Figure 3.

d. Experiment 4 - Impact of Command and Control Uncertainties on the Periodicity of the Fundamental Update Interval Starts.

This experiment involved the use of a GASP-based dynamic model. The results are illustrated by a GASP histogram plot. For this experiment, a bus load which consisted primarily of the periodic messages plus background demand messages, and a command and control scheme which consisted of addressing a terminal and waiting for a terminal to respond, were assumed. The delays in terminal response could conceivably cause the start of an update interval to be delayed until a response is received from a terminal. The response time variations are attributed to several factors, including clock variation and problem-caused variations such as failures of the response mechanism, whereas the bus controller has to await the timing out of a watchdog timer, which disables the terminal from responding, before the bus controller is free to proceed. This delay can cause the scheduled start time for the next message to be delayed, thereby impacting the start of the next update cycle or fundamental update interval. Should this happen, it delays the start of every message in that interval by that amount.

The histogram in Figure 4 shows statistics of the update interval start time jitter (expressed in fractions of the update interval duration) measured while running this particular experiment.

4. CONCLUSIONS

In summary, it was stated that MUXSIM in its present form can provide immediate and significant benefits to both the multiplex system designer and to associated personnel, who may be production engineers or software developers. Also, four specific simulation experiments were described together with results to illustrate some particular simulation experiments that can be easily performed by a MUXSIM Basic User. Although these experiments are thought to be typical of those likely to be performed by a multiplex system designer, no extensive analyses were made of the results, and no hard conclusions regarding multiplex systems being simulated could be drawn. The phenomena explored included bus loading and saturation effects, and the impact of redundancy management schemes.

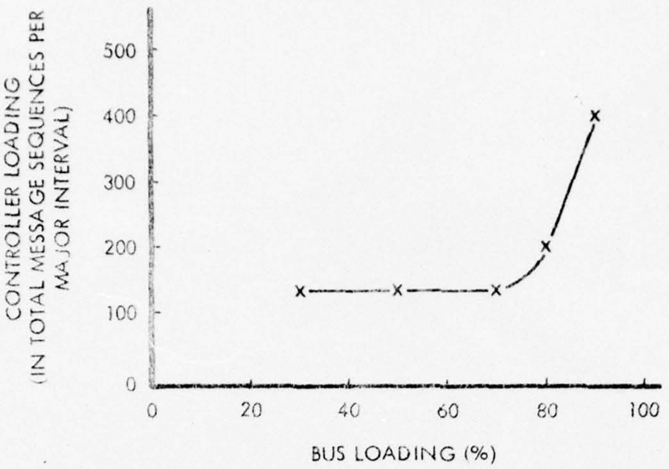
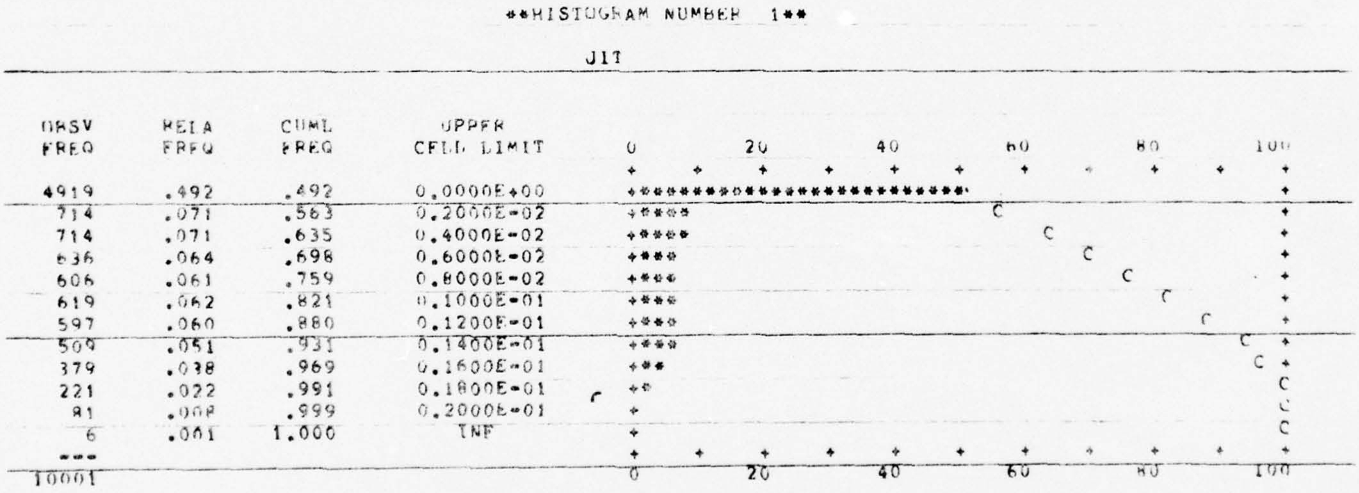


Figure 3. Controller Loading Versus Bus Loading



```
                              **HISTOGRAM NUMBER  1**

                                          JIT

OBSV    RELA    CUML       UPPER
FREQ    FREQ    FREQ    CELL LIMIT    0         20        40        60        80        100
                                      +    +    +    +    +    +    +    +    +    +    +
4919    .492    .492    0.0000E+00    +************************                         +
 714    .071    .563    0.2000E-02    +****                       C                     +
 714    .071    .635    0.4000E-02    +*****                          C                 +
 636    .064    .698    0.6000E-02    +***                               C              +
 606    .061    .759    0.8000E-02    +***                                  C           +
 619    .062    .821    0.1000E-01    +***                                     C        +
 597    .060    .880    0.1200E-01    +***                                        C     +
 509    .051    .931    0.1400E-01    +***                                           C  +
 379    .038    .969    0.1600E-01    +**                                             C +
 221    .022    .991    0.1800E-01 C  +*                                                C
  81    .008    .999    0.2000E-01    +                                                 C
   6    .001   1.000       INF        +                                                 C
 ---                                  +    +    +    +    +    +    +    +    +    +    +
10001                                 0         20        40        60        80        100
```

Figure 4. Statistics of Update Interval Start Time Jitter

SECTION V

MUXSIM USE

1. INTRODUCTION

This section is primarily addressed to the two questions, "How is MUXSIM used?" and "What are MUXSIM use costs?", for the typical Basic User. The first question was briefly treated in Section III(MUXSIM Description); here it is covered in somewhat more detail. The second question is partially answered by reference to computer resource costs associated with the conduct of the four simulation experiments discussed in Section IV, where the question was treated briefly.

2. USER TYPES

As was noted earlier, there are two main types of MUXSIM user: the Basic User and the Advanced User. The former uses MUXSIM essentially as is and relies primarily on the MUXSIM User's Manual, together with the built-in coaching features of the MUXSIM Executive, in order to conduct his simulation exercises. The latter requires a more detailed knowledge of MUXSIM software. Therefore, he generally requires information found in the MUXSIM System Modification Design Data Manual. He may also require knowledge of Fortran, GASP, and/or TOPS-10, depending on the type of MUXSIM use he has in mind. Both types of user are assumed to be familiar with the elements of multiplex system design. This section summarizes the needs and procedures of the MUXSIM Basic User, while Section VI summarizes needs and procedures of the Advanced User.

3. TYPICAL MUXSIM USE

To discuss how MUXSIM is typically used, it is first necessary to define a typical MUXSIM use. For present purposes, we assume that a typical MUXSIM use is the conduct of a simulation experiment such as described by the flow chart of Figure 5. Given this flow chart, the Basic User may be defined more precisely as one who normally uses all of the blocks shown except Block 6.

4. BASIC MUXSIM USE

The starting point for any employment of MUXSIM, whether by the Basic User or Advanced User, is the determination of the multiplex system design question for which an answer is desired. Depending on what this question is, MUXSIM may or may not be applicable. (Here, "applicable" means that MUXSIM can be either directly used without modification, or that the required MUXSIM modifications are compatible with the user's time and budget constraints). If MUXSIM is not applicable, then some other analysis technique must be selected and used in order to answer the question. Blocks 1, 2 and 3 of the flow chart are concerned with these matters, and at present all the steps involved are conducted manually. The user must be sufficiently cognizant of the nature and structure of MUXSIM to carry out the steps satisfactorily.

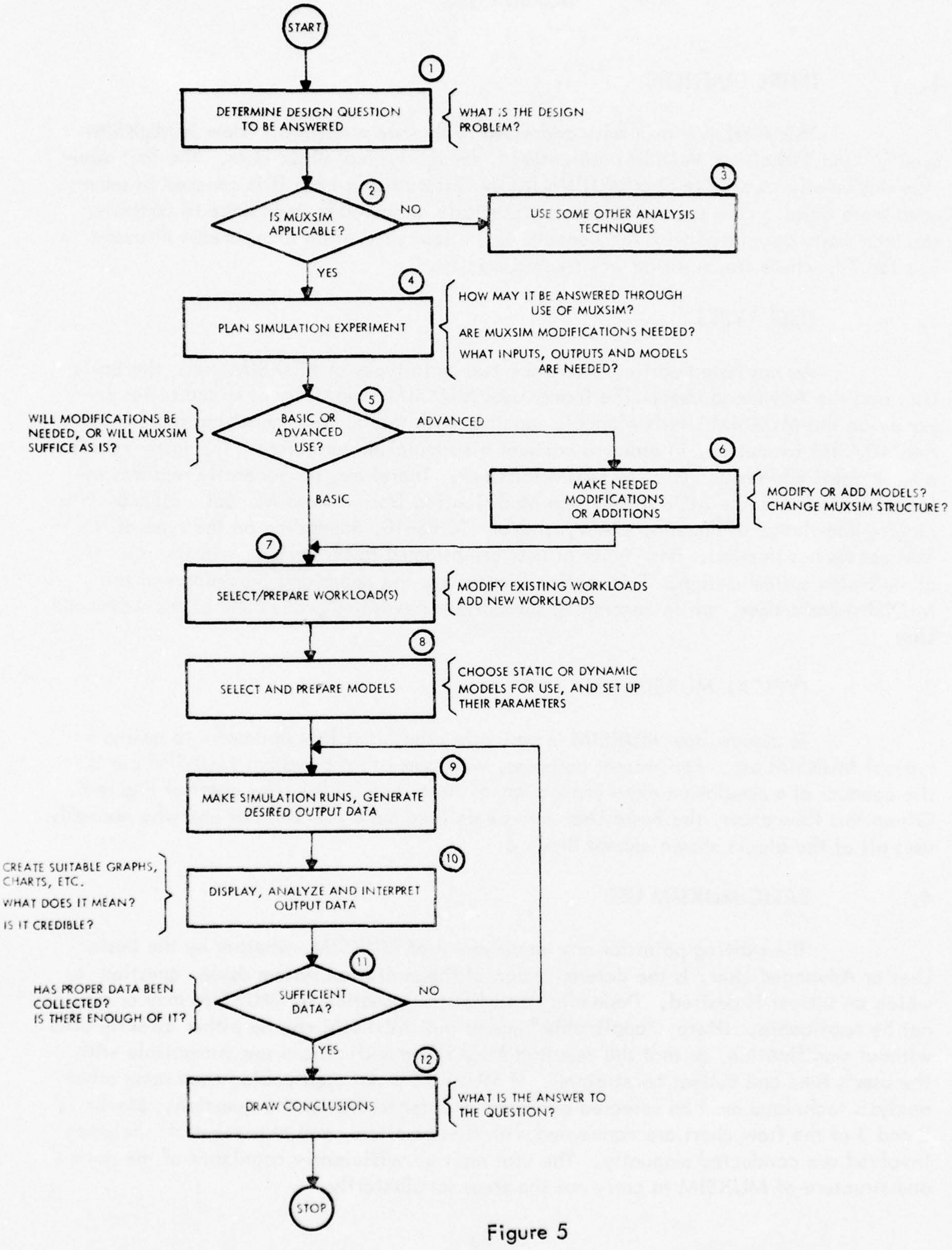

START
1
DETERMINE DESIGN QUESTION TO BE ANSWERED
WHAT IS THE DESIGN PROBLEM?
2
IS MUXSIM APPLICABLE?
NO
3
USE SOME OTHER ANALYSIS TECHNIQUES
YES
4
PLAN SIMULATION EXPERIMENT
HOW MAY IT BE ANSWERED THROUGH USE OF MUXSIM?
ARE MUXSIM MODIFICATIONS NEEDED?
WHAT INPUTS, OUTPUTS AND MODELS ARE NEEDED?
WILL MODIFICATIONS BE NEEDED, OR WILL MUXSIM SUFFICE AS IS?
5
BASIC OR ADVANCED USE?
ADVANCED
6
MAKE NEEDED MODIFICATIONS OR ADDITIONS
MODIFY OR ADD MODELS? CHANGE MUXSIM STRUCTURE?
BASIC
7
SELECT/PREPARE WORKLOAD(S)
MODIFY EXISTING WORKLOADS
ADD NEW WORKLOADS
8
SELECT AND PREPARE MODELS
CHOOSE STATIC OR DYNAMIC MODELS FOR USE, AND SET UP THEIR PARAMETERS
9
MAKE SIMULATION RUNS, GENERATE DESIRED OUTPUT DATA
CREATE SUITABLE GRAPHS, CHARTS, ETC.
WHAT DOES IT MEAN?
IS IT CREDIBLE?
10
DISPLAY, ANALYZE AND INTERPRET OUTPUT DATA
HAS PROPER DATA BEEN COLLECTED?
IS THERE ENOUGH OF IT?
11
SUFFICIENT DATA?
NO
YES
12
DRAW CONCLUSIONS
WHAT IS THE ANSWER TO THE QUESTION?
STOP

Figure 5

Typical MUXSIM Use – Conduct of a Simulation Experiment

Block 4 involves the planning of the simulation experiments. For the most part, the detailed steps required are similar to those needed in the planning of any scientific experiment, whether it be simulation, physical, or other. Typical considerations include preparation of inputs, provision for outputs, length of time required for the experiment, statistical validity, errors introduced by instruments, etc. These subjects will not be covered in detail here - suffice it to say that, for this typical use, the MUXSIM user is expected to have a basic understanding of the scientific method and an ability to map normal good experimental practice into the MUXSIM context. Like the other blocks discussed above, Block 4 is mainly implemented by manual means.

Blocks 7, 8, 9, and 10 in the simulation experiment flow chart are those which are mainly mechanized by MUXSIM. It is here that the four major MUXSIM subsystems (Utility, Static, Dynamic, and Executive) come into play. Briefly, the Utility subsystem provides a library of preprepared workloads and a set of aids for workload development and modification; the Static subsystem provides a library of preprepared "steady-state" multiplex system models; the Dynamic subsystem provides preprepared stochastic models; and the Executive subsystem controls operations of the other subsystems while also providing powerful coaching features for the user if he selects this option. It is largely this collection of MUXSIM features which substantiates the claim that, for the Basic User, MUXSIM is, indeed, "easy to use." Those who have used any modern computer time-sharing system will readily understand the general nature of these MUXSIM user-oriented features. Their details and specifics are given in the MUXSIM User's Manual and are not repeated in this Final Report.

Block 10 of the flow chart is largely accomplished by manual means. It is at this point in a simulation experiment that the Basic User decides whether or not he has sufficient data, whether the data he has seems credible, etc. For example, if he is using the output data to plot a curve (of the results of the simulation experiments discussed earlier), he may find it necessary to obtain additional data points to better display the shape of the curve, especially for its critical regions. Finally, block 11 is where the experimenter attempts to pull together all simulation results in order to draw a conclusion and (hopefully) to either answer or shed light on his original question. This, too, is largely a manual operation at present.

5. COSTS OF MUXSIM USE

Although simulation can produce interesting and useful results, it is not uncommon to find inappropriate uses of simulators wherein trivial results are generated at substantial computer costs, or where useful results are produced but at very high computer costs. An example of the latter could be the estimation of multiplex bus bit error rates via simulation; in this case, the desired estimates might be obtained after much computer number crunching, while estimates of comparable accuracy could have been obtained by other means (e.g., by manual analysis or by breadboard techniques) at considerably lower costs.

In the specification and design of MUXSIM, a strong effort was made to avoid its use in areas of predictable low cost-effectiveness. It is hoped that, for the most part, this effort has been successful; however, the overall cost-effectiveness of MUXSIM cannot be rigorously demonstrated without a significant operational evaluation effort. This has not yet been completed. Therefore, the best that can be done at this point is to show some indications of MUXSIM cost-effectiveness. That is the purpose of this section.

The method used to obtain these indications was to consider some typical simulation experiments carried out using MUXSIM, and to estimate the computer costs associated with running each experiment. These costs may be estimated by measuring the computer resources used for each experiment, and by assigning costs for these resources using costing schemes consistent with industry practice; i.e., by using the "going rate" for the various computer resources used. The first task is not difficult, since most operating systems for large computer systems like the DEC System-10 produce computer resource utilization accounting data as a by-product of normal operations.

Applying this costing methodology to the four simulation experiments described earlier, results are as shown in Table II (Section III). From this data is may be inferred that a "typical" MUXSIM *simulation experiment* has a direct computer resource cost of about $50-$200 based on experiment run time and CPU cost of $8/min. Of course, though, this figure will vary widely with computer resource accounting practices from one host computer system to another, and so it is useful as a "ball park" estimate only.

6. SUMMARY AND CONCLUSIONS

This section has been concerned with the two questions "How is MUXSIM used?" and "What are MUXSIM use costs?", for the typical Basic User. In order to get at the answer to these questions, a typical MUXSIM use was defined by a flow chart which covers the case of a routine simulation experiment. The flow chart serves to show the simulation experiment as a whole; it also shows specifically how and where MUXSIM aids in the conduct of the experiment. Certain steps must be accomplished manually. Examination of some of these steps serves to show what requirements are placed on the user in order for him to use MUXSIM effectively.

Costs of "typical" simulation experiments are estimated by conventional means, and found to be $50-$200 for the simulation experiments 1 through 4 described in Section IV. These estimates are intended to be considered as "ball park" figures only, since the costs will vary with different simulation experiments and with different computer resource accounting methods. The cost-effectiveness of MUXSIM is not, therefore, rigorously demonstrated by this analysis effort; rather, it is shown that costs of certain simulation experiments felt to be typical are not excessive.

In conclusion, results of the analysis efforts reported here indicate that MUXSIM should be both easy to use and cost-effective for the Basic User. However, firm conclusions on this matter must await a more complete MUXSIM operational evaluation.

SECTION VI

MUXSIM MODIFICATION

The Advanced User may wish to modify MUXSIM by alteration or addition of Static and Dynamic models and in many other possible ways. In addition, others may wish to modify MUXSIM for various reasons; for example, it may be desired to move MUXSIM to a new host system, to enhance MUXSIM with features conceptually defined but not yet implemented, to add more coaching features to the Executive, to decrease MUXSIM run times by tuning operations, etc. For those wishing to modify MUXSIM for any reason, the MUXSIM System Modification Design Data Manual is the primary source of information.

A MUXSIM modification may be very easy or quite difficult, depending on the type desired. Factors tending to make the modification easier include the following:

- MUXSIM is implemented almost entirely in Fortran IV and GASP.
- MUXSIM is implemented in modular fashion, with relatively clean intermodule interfaces.
- MUXSIM is well documented in the System Modification Design Data Manual.

However, certain modifications, such as those needed to move MUXSIM to another host computer system, can be fairly complex since portions of MUXSIM (mainly the "user-convenience" software) are operating system-dependent. This is because, to reduce MUXSIM development costs, certain MUXSIM functions are implemented using TOPS-10 (the DEC System-10 operating system in use at AFAL) instead of making MUXSIM completely self-contained. Thus, when MUXSIM is moved to a new operating system, these functions must be re-implemented either as part of MUXSIM itself or through the use of suitable functions of the new operating system.

Whatever the MUXSIM modification desired, the information necessary to perform it is contained in the MUXSIM System Modification Design Data Manual, which includes sections on MUXSIM architecture, components, modification, and installation. It also contains the MUXSIM functional specification.

Recommended Procedures for certain specific types of modifications are given explicitly, while for other types the modifier must plan his own procedures. The reader interested in details concerning this subject is referred to the System Modification Design Data Manual for further information.

SECTION VII

MUXSIM HISTORY

1. OVERVIEW

MUXSIM history is covered in detail in Appendices A and B of this report. Accordingly, the purpose of this section is to give only a brief summary of that history. Overall, the MUXSIM design and development effort has been a three-phase, 31-month effort. The three phases are discussed separately in the following subsections.

2. MUXSIM PHASE I - DEFINITION

Phase I was a definition study which addressed the design alternatives for a multiplex system simulator capable of permitting the evaluation of data transfer concepts and techniques. It culminated in a two-volume Interim Technical Report (Ref. 5) which describes the elements of air vehicle multiplex systems and the concepts required to simulate them. The referenced report contains three main sections, embodying coverage of multiplex systems analysis, simulator inputs/outputs and variables, and simulator construction methods.

The most significant question arising during Phase I involved the clear definition of simulator outputs. In the study it was concluded that, for maximum Air Force benefits, MUXSIM should be designed so as to be applicable at the earliest possible time *in the multiplex system design cycle.* MUXSIM outputs were selected so as to be compatible with that design objective. Other Phase I recommendations based on the study results were as follows:

- Certain aspects of multiplex system design are best handled on an analytic basis. These include bus-related matters such as bit error rate, impedance matching and transmission line problems, TDM vs FDM trade-off studies, etc. Although computer programs can be valuable in support of such studies, it was recommended that such programs not be made part of MUXSIM initially in order to avoid the problems of excessive simulator run times and excessive simulator size and development costs.

- To reduce MUXSIM initial development costs and facilitate later changes, the highest level implementation language feasible should be used. Specifically, GASP and/or Fortran should be used as primary implementation languages for this simulator.

- The test-bed mode of MUXSIM operation, with its real-time implications, should be considered as separate from and independent of other modes of operation.

- The non-test bed portion of MUXSIM should be considered mainly as software, with little or no special hardware required.

- To ensure MUXSIM feasibility and relevance, MUXSIM specifications should be developed in parallel with a small-scale simulation exercise to model and simulate those areas of the B-1 EMUX System which have in hindsight caused the most design problems, and wherein early simulation could apparently have done the most to alleviate those problems. An example of such a problem area is that of redundancy management. MUXSIM specifications should be developed in part by generalization of the small-scale simulation model developed for the B-1 EMUX analysis.

- The MUXSIM project should be continued through specification and prototype development phases, because there is strong evidence that MUXSIM can be a very powerful and cost-effective design tool.

More details regarding specific Phase I study methodology, results, and conclusions may be found in Appendix A and in Reference 5.

3. MUXSIM PHASE II DESIGN

Phase II was a follow-on study leading to the functional design and specification of MUXSIM. Results, conclusions, and recommendations of that study are contained in the four-volume Second Interim Technical Report (Ref. 6). A condensed and revised version of that report is included in Appendix A to this report.

The main objective of the MUXSIM Phase II study was the specification of MUXSIM design, together with a discussion of the design features and rationale. The resulting design was consistent with the results and recommendations of the Phase I study without any major differences, except that DAIS was used as the object of a small-scale probe-coding exercise instead of B-1 EMUX.

A major MUXSIM design goal was cost-effectiveness. This was pursued, in general, by attempting to keep MUXSIM development and use costs low while maximizing use benefits. The attempt to maximize benefits involved primarily a continuing emphasis on MUXSIM focus and scope, realism, ease of use, and flexibility. Substantially, this meant:

1. Focus on those practical and important multiplex system design questions which must be answered early in the multiplex system design cycle.

2. Focus on design questions which are best answered by simulation.

3. Emphasis on real multiplex system workloads.

4. Emphasis on features which will make MUXSIM a tool which is convenient and easy to use.

5. Emphasis on features which make MUXSIM easy to modify and/or expand (i.e., flexibility features).

The single most important factor in maximizing MUXSIM benefits was the heavy involvement in MUXSIM design of experienced multiplex system design engineers who were very familiar with what design questions must be answered earliest, which are the most important, and which cannot be answered easily by other available tools and techniques.

The attempt to minimize MUXSIM development and use costs involved a number of approaches, including the following:

- Keeping software development costs down by use of high-level implementation language (e.g., by the use of Fortran).
- Keeping software development costs down by avoiding "re-invention of the wheel" (e.g., by use of GASP for development of dynamic simulation models).
- Keeping software development costs down (and flexibility up) by use of modular software.
- Keeping software development costs down by maximum use of functions provided by the host system's operating system.
- Keeping development and use costs down by making MUXSIM itself as small as possible initially (this relates directly to the scope issue mentioned earlier).
- Keeping use costs down by making MUXSIM interactive with extensive coaching features (this cuts down significantly on user training costs and on user time, but not necessarily on computer time components of use cost).

The single most important factor in arriving at a MUXSIM design for minimum development and use costs was the heavy involvement of software designers who were experienced in the design and use of interactive simulation packages.

As part of the Phase II effort, a MUXSIM system specification was prepared using guidelines set forth in MIL-STD-490 and MIL-STD-483. Additional detail was supplied from a number of other documents, some of which were derived from probe-coding exercises which were done as part of this study phase. Details of this data are in Appendix A of this report, and in Ref. 6.

4. MUXSIM PHASE III - IMPLEMENTATION AND VERIFICATION

Phase III was the actual implementation and verification phase of MUXSIM. It has resulted in an operational version of MUXSIM, together with all required supporting documentation. The latter includes this Report as well as the MUXSIM User's Manual and the MUXSIM System Modification Design Data Manual. A GASP IV Manual (ref. 4) is also needed. (This GASP IV manual is available from almost any good technical bookstore.) Since a detailed report on MUXSIM Phase III is included as Appendix B to this Report, only a summary and highlights of it are given here.

For the most part, MUXSIM was implemented and verified according to the specifications and plans set forth as the outputs of Phase II of the Study. The two main changes to the original specifications were the following:

1. The MUXSIM host system was redesignated to the DEC System-10 instead of the originally planned PDP-11/45.

2. A number of MUXSIM output formats were changed so as to be more user-oriented than those originally planned.

The first-mentioned change required some MUXSIM redesign because parts of MUXSIM (mainly the "convenience" software) are operating system-dependent. The second-mentioned change entailed mainly the addition of dictionaries to allow MUXSIM outputs to be more in text form and less in terms of numbers.

As now implemented, MUXSIM comprises four major subsystems: the Utility, the Static, the Dynamic, and the Executive. These subsystems are in turn divided into programs, then subprograms, and finally into subroutines or modules. Most code is written in Fortran, but some is in GASP, and the Executive makes some use of the TOPS-10 control statements. For development purposes, code testing was done at the module level; but for MUXSIM verification purposes, testing was done at a program level. As was noted earlier, four small simulation experiments were conducted as part of the MUXSIM verification effort. All necessary details regarding MUXSIM as implemented may be found in the MUXSIM System Modification Design Data Manual, while design rationale will be found in Appendixes A and B to this report as well as in References 5 and 6.

SECTION VIII

MUXSIM FUTURE

This section is concerned mainly with the question "Where should (or might) MUXSIM go from here?" In treating this question, some Harris opinions on the matter are presented, together with a listing of some areas where MUXSIM improvements or enhancements might now be desirable. The latter are derived from Harris' experiences in the implementation, verification, and limited operational use of MUXSIM.

To review the situation briefly, MUXSIM is a multiplexer system designer's tool which has been designed, implemented, and verified; but it is a tool which has not yet been extensively tested through actual use. Therefore, it is Harris' opinion that the next logical step is the operational evaluation of MUXSIM, and that such an evaluation will provide the best possible source of inputs regarding needed improvements and enhancements, if any.

Harris believes strongly that in the case of a tool such as MUXSIM, evolutionary improvements based solidly on its use history are the best kind. It is noteworthy that a number of successful, modern software simulation packages have been developed by this route; SIMSCRIPT, GASP, and GPSS are cases in point.

Although real use generally is the best source of information on MUXSIM development enhancements that may be needed, there are some areas where the need for further development is already quite clear. For example, a larger workload library is evidently desirable. Enhancements like this one require no changes to MUXSIM structure, and immediately extend the range of useful results that can be derived from MUXSIM.

Other specific possibilities for MUXSIM development, which emerged as Harris took MUXSIM through implementation and verification stages, include the following:

- Use of a graphics terminal to aid in remote terminal assignments.
- Better integration of MUXSIM dynamic models with real world data bases (workloads).
- Further growth of dynamic models; e.g., to allow direct investigation of multiplex processor and bus controller loading phenomena.

- Further development of static models to allow study of multiple bus and multi-line bus configurations.

Many other possibilities could be mentioned in addition to the above. However, consistent with the view that further development needs are best established through real use experience, Harris recommends implementation of them on an "as need" basis.

SECTION IX

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

This Report has covered the design, development, implementation, and verification of a multiplex system simulator called MUXSIM. MUXSIM is intended as a multiplex system designer's analysis tool, to be used primarily in the early stages of a multiplex system development cycle. As an analysis tool, it was intended to be easy to use, flexible, and productive of meaningful and credible results. Above all, it was intended to be cost-effective.

Following the Introduction, this Report has included sections covering MUXSIM purpose, description, benefits, use, modification, history, and its future. In all cases, the coverage has been in summary fashion with supporting details left either for the Appendixes or for two other study-generated documents, namely, the MUXSIM User's Manual and the MUXSIM System Modification Design Data Manual.

The main overall conclusions are that MUXSIM was built per spec, its correct functional operation has been verified, and it is thought to be cost-effective. It is obviously neither a perfect tool nor as complete as it might be, but it is designed to be flexible so as to allow for easy modification and/or growth.

The best recommendation that Harris makes at this time is that further operational evaluation of MUXSIM be conducted; i.e., that it be put to the test in a real multiplex system design context. Hopefully, such a test will reveal that MUXSIM is truly useful and beneficial. Also, such a test is viewed as the best possible source of inputs for further MUXSIM development needs.

APPENDIX A

MUXSIM DEFINITION AND DESIGN

PHASE I AND PHASE II REPORT

SECTION I

INTRODUCTION

The purpose of this appendix is to provide supporting detail regarding procedures, results, and rationale related to the definition and design phases of MUXSIM development. Most of the data contained herein was obtained by the editing and reorganization of data contained in the Phase I and Phase II Interim Reports (references 5 and 6). The interested reader can find additional information on these subjects in those two reports. The areas primarily supported by details in this appendix are MUXSIM Description and MUXSIM History.

The remainder of this appendix is divided into five sections. Section II covers the overall MUXSIM development plan, including its relation to specific statement of work tasks. Section III covers Phase I (MUXSIM Definition), and includes details on the what and why of MUXSIM description from a functional or user's viewpoint. Section IV covers Phase II (MUXSIM Design), and includes details on the what and why of MUXSIM description from a logical or implementer's viewpoint. Both Sections II and IV contain historical information as appropriate, mainly concerning how the particular description in question was obtained. Finally, Section V provides a summary and conclusions for the MUXSIM Definition and Design phases.

SECTION II

MUXSIM DEVELOPMENT PLAN

To put this appendix in perspective, the overall MUXSIM development plan is shown in bubble chart form in Figure A-1. As has been noted previously, this appendix contains details of the what, why, and how of Phases I and II only. Appendix B gives a similar treatment of Phase III, while Phase IV has not yet been accomplished. In addition to identifying the phases and their sequence, Figure A-1 shows the principal outputs of each Phase.

A Statement of Work (SOW) was used by Harris as a guide in carrying out the development plan. Altogether, eleven SOW tasks were defined and executed in the course of MUXSIM development. These tasks are identified and their relations to the MUXSIM phases are shown by Table A-1 and Figure A-2.

Details of the what, why, and how of the MUXSIM definition and design phases are given in the next two sections.

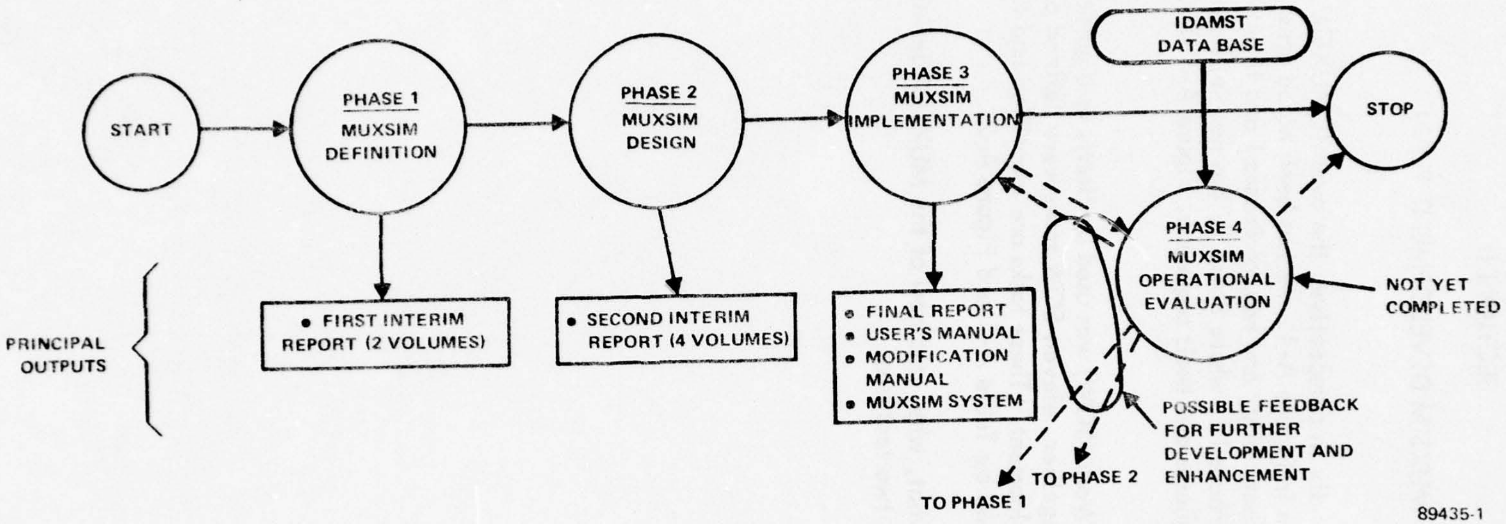


Figure A-1.

MUXSIM Development Plan

Table A-1.

FLOW BLOCK- SOW TASK INTERRELATIONS

SOW TASKS / MAJOR STUDY FLOW BLOCKS		MUX System Parameters	MUX System Parameter Formats	Variables of Simulation	Simulator Outputs	MUX System General Functions	MUX System General Function Implementation	Simulator Control System	Simulator Detailed Design	Real-Time/ Simulation Time Relationship	Simulator Inputs	Simulator Implementation
		1	2	3	4	5	6	7	8	9	10	11
1. Formalize Multiplex System Design process	PHASE 1	●				●						
2. Identify MUX System Design questions for which analysis by simulation is most cost-effective		●		●	●	●				●	●	
3. Perform functional design of a Multiplex System Simulator	PHASE 2			●	●	●		●				
4. Perform detail design of a Multiplex System Simulator		●	●	●	●	●	●	●	●	●	●	
5. Implement a Multiplex System Simulator Design	PHASE 3											●

1. A black dot in the matrix indicates a strong relationship between the SOW Task and the Study Flow Block.

2. SOW Task definitions are as follows:

- Task 1 - Develop techniques for specifying multiplex system parameters.
- Task 2 - Specify formats for multiplex system parameters.
- Task 3 - Specify variables of simulation.
- Task 4 - Specify simulator outputs.
- Task 5 - Specify generalized functions which comprise a multiplex system.
- Task 6 - Specify methods for implementation of multiplex system generalized functions.
- Task 7 - Specify multiplex system simulator control system.
- Task 8 - Specify simulator detailed design.
- Task 9 - Specify real-time/ simulated-time relationships.
- Task 10 - Specify simulator inputs and methods for their generation.
- Task 11 - Implement the simulator design and verify the implementation.

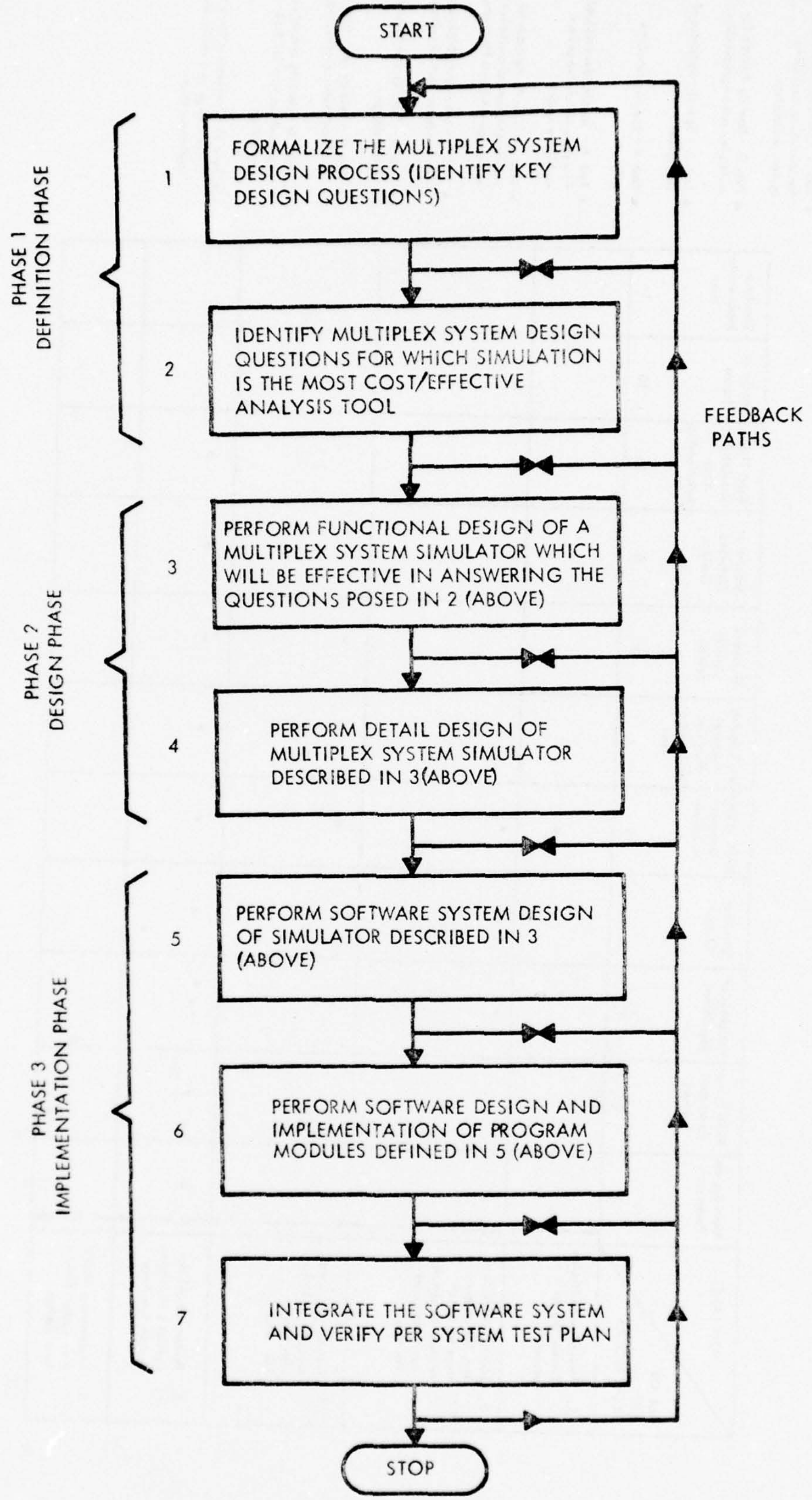

START
PHASE 1
DEFINITION PHASE
1
FORMALIZE THE MULTIPLEX SYSTEM DESIGN PROCESS (IDENTIFY KEY DESIGN QUESTIONS)
2
IDENTIFY MULTIPLEX SYSTEM DESIGN QUESTIONS FOR WHICH SIMULATION IS THE MOST COST/EFFECTIVE ANALYSIS TOOL
FEEDBACK PATHS
PHASE 2
DESIGN PHASE
3
PERFORM FUNCTIONAL DESIGN OF A MULTIPLEX SYSTEM SIMULATOR WHICH WILL BE EFFECTIVE IN ANSWERING THE QUESTIONS POSED IN 2 (ABOVE)
4
PERFORM DETAIL DESIGN OF MULTIPLEX SYSTEM SIMULATOR DESCRIBED IN 3(ABOVE)
PHASE 3
IMPLEMENTATION PHASE
5
PERFORM SOFTWARE SYSTEM DESIGN OF SIMULATOR DESCRIBED IN 3 (ABOVE)
6
PERFORM SOFTWARE DESIGN AND IMPLEMENTATION OF PROGRAM MODULES DEFINED IN 5 (ABOVE)
7
INTEGRATE THE SOFTWARE SYSTEM AND VERIFY PER SYSTEM TEST PLAN
STOP

Figure A-2. Simplified Development Flow Chart

SECTION III

PHASE I - MUXSIM DEFINITION

1. INTRODUCTION

A summary view of the four major task groups which were performed in the definition phase is shown in bubble chart form in Figure A-3. The various bubbles are annotated to give some idea of the associated work details involved. A statement of the general goal of the overall development effort is also shown in this figure, together with a major criterion applicable to that goal.

As may be seen from the above, both the goal and its major criterion are quite general. This fact accounts for the major problem in Phase I, which was to define the problem. Many possible simulation tools could be defined which satisfy the general goal, and a fairly large subset of these could be cost-effective. The MUXSIM definition phase can be characterized as a complex multi-stage screening process, the main result of which was the scoping of MUXSIM. Details of the program and its results are given in the following sections. Each of the task groups shown in Figure A-3 is covered in a separate section.

2. TASK GROUP 1 - FORMALIZE MUX DESIGN PROCESS

If a good multiplex system designer's handbook existed, it would have been unnecessary to perform this task group; but since one did not exist, its equivalent had to be created. The results of the pseudo-handbook development effort are contained in full in Volume II of the first Interim Report (ref. 5).

First Interim Report - Volume II is a comprehensive, 118-page document which includes a complete catalog of the elements of multiplex systems, a general description of FDM concepts, a description of TDM techniques, and another catalog of alternative command and control implementations. Although it was created mainly to provide the background information necessary for simulator definition efforts, it can also be used on a stand-alone basis to provide insight into information-transfer techniques. In fact, the document as it now exists could be used as a starting point for the multiplex system designer's handbook mentioned earlier.

To give an idea of the details covered in this document, its Table of Contents is given here as Figure A-4. As may be seen, there are three main substantive sections, covering multiplex system considerations, structures, and command and control techniques, respectively. The first of these includes a catalog of multiplex system considerations, a condensed version of which is given here as Figure A-5. The catalog is given in outline form only; no attempt was made to fill in the details.

The bus structures section covers the two most commonly used multiplexing techniques, namely Frequency Division Multiplex (FDM) and Time Division Multiplex (TDM). These techniques are discussed in a tutorial sense to provide the reader with an

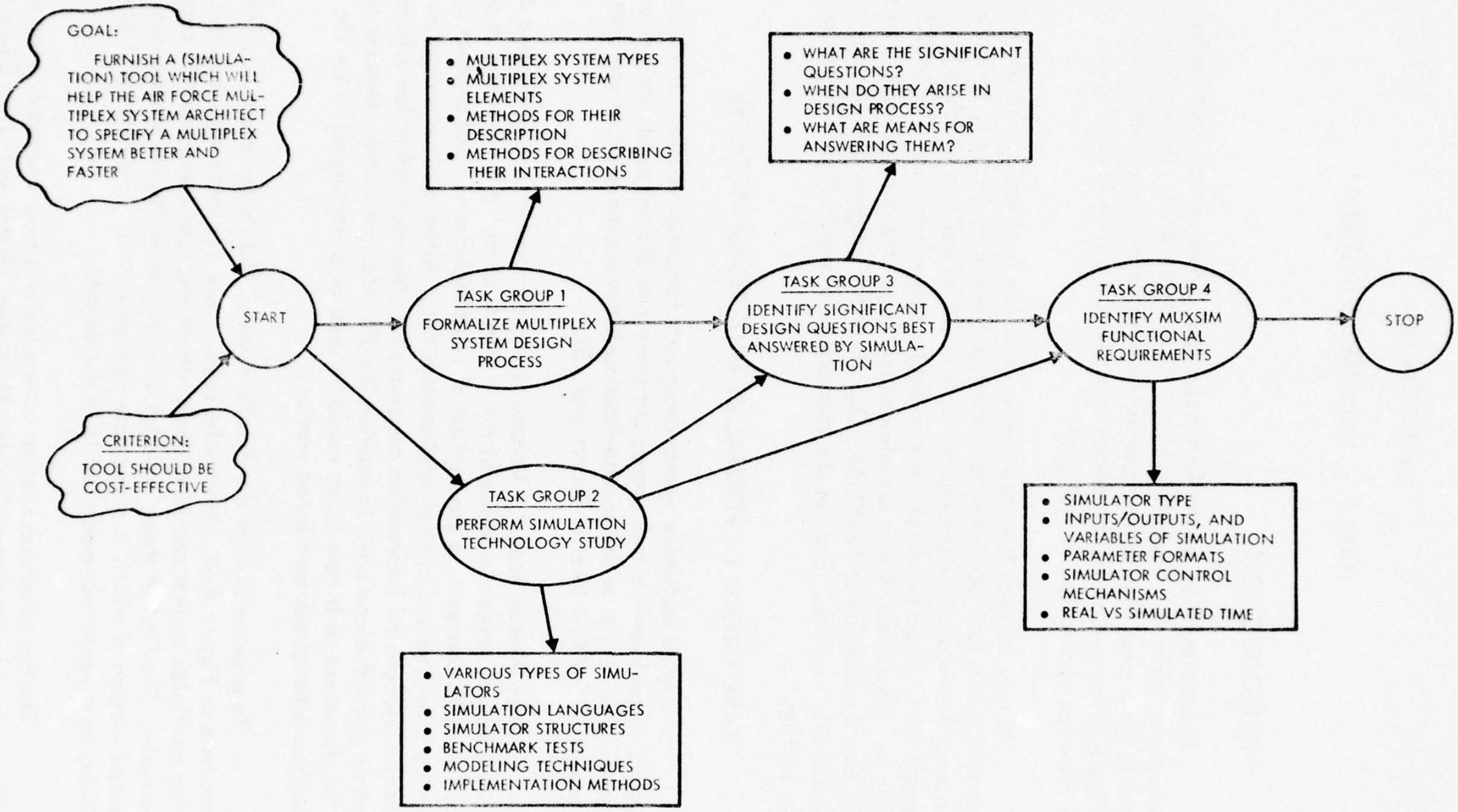

GOAL:
FURNISH A (SIMULATION) TOOL WHICH WILL HELP THE AIR FORCE MULTIPLEX SYSTEM ARCHITECT TO SPECIFY A MULTIPLEX SYSTEM BETTER AND FASTER
CRITERION:
TOOL SHOULD BE COST-EFFECTIVE
START
TASK GROUP 1
FORMALIZE MULTIPLEX SYSTEM DESIGN PROCESS
• MULTIPLEX SYSTEM TYPES
• MULTIPLEX SYSTEM ELEMENTS
• METHODS FOR THEIR DESCRIPTION
• METHODS FOR DESCRIBING THEIR INTERACTIONS
TASK GROUP 2
PERFORM SIMULATION TECHNOLOGY STUDY
• VARIOUS TYPES OF SIMULATORS
• SIMULATION LANGUAGES
• SIMULATOR STRUCTURES
• BENCHMARK TESTS
• MODELING TECHNIQUES
• IMPLEMENTATION METHODS
TASK GROUP 3
IDENTIFY SIGNIFICANT DESIGN QUESTIONS BEST ANSWERED BY SIMULATION
• WHAT ARE THE SIGNIFICANT QUESTIONS?
• WHEN DO THEY ARISE IN DESIGN PROCESS?
• WHAT ARE MEANS FOR ANSWERING THEM?
TASK GROUP 4
IDENTIFY MUXSIM FUNCTIONAL REQUIREMENTS
• SIMULATOR TYPE
• INPUTS/OUTPUTS, AND VARIABLES OF SIMULATION
• PARAMETER FORMATS
• SIMULATOR CONTROL MECHANISMS
• REAL VS SIMULATED TIME
STOP

Figure A-3.

MUXSIM Definition Phase

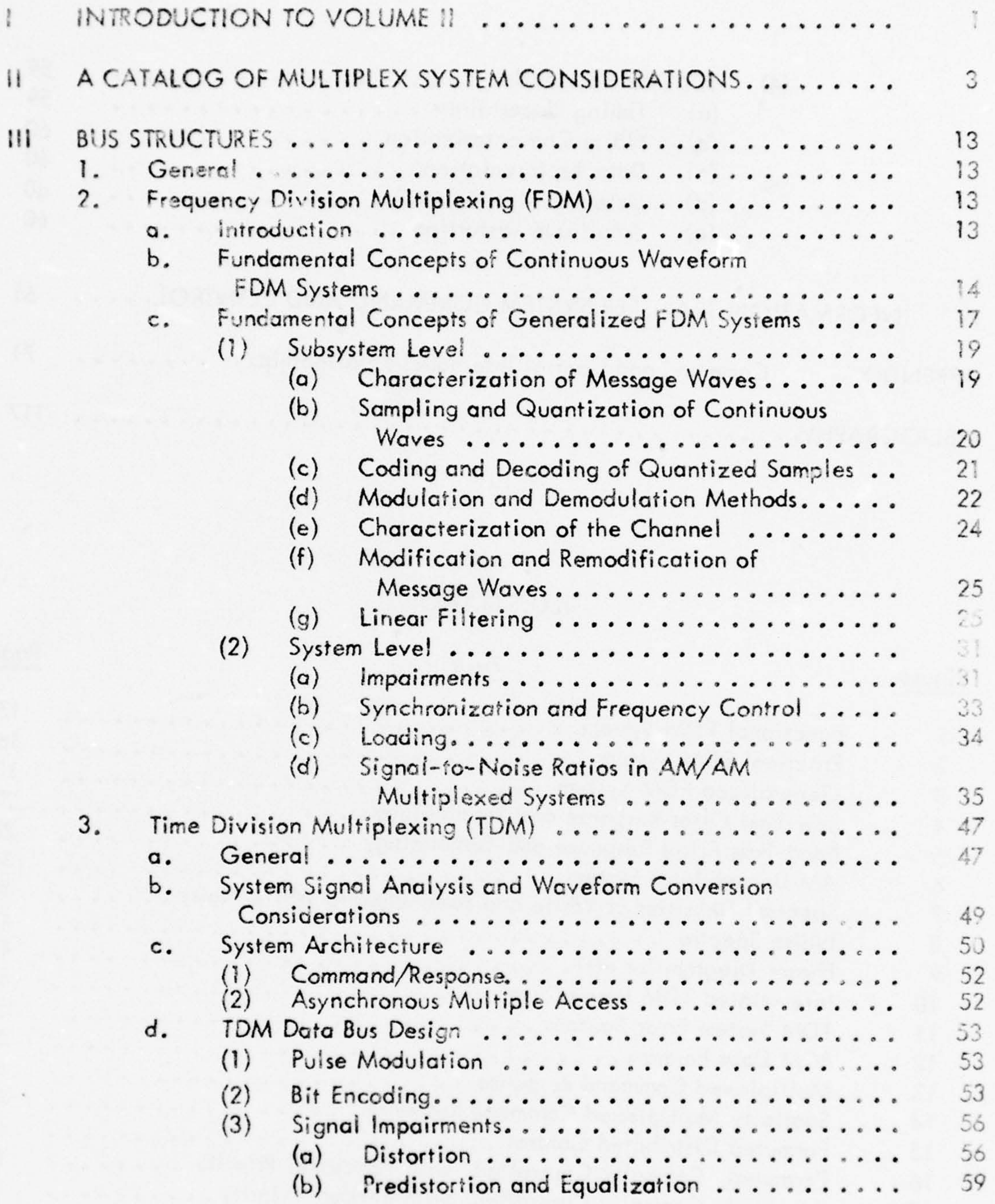


Figure A-4. Table of Contents, Volume II - First Interim Technical Report (Sheet 1 of 2)

ILLUSTRATIONS

1.0 SYNCHRONIZATION, TIMING AND CONTROL

1.1 Information Routing
1.2 Control Architecture
1.3 Bus Access
1.4 Timing and Synchronization
1.5 Information Transfer Format Design
1.6 Terminal Processing
1.7 Queueing (Resource Sharing and Contention)

2.0 SIGNAL DESIGN AND DETECTION

2.1 Channel Isolation
2.2 Time Division Multiplexing
2.3 Frequency Division Multiplexing
2.4 Code Division Multiplexing
2.5 Message Format Design
2.6 Error Correction and Detection Coding
2.7 Bit Encoding Formats
2.8 Special Encoding
2.9 Data Transfer
2.10 Processing

3.0 TRANSMISSION MEDIA

3.1 Transmission Media Types
3.2 Wireline Characteristics
3.3 Wireline Types
3.4 Operating Modes
3.5 Design Factors

Figure A-5. Condensed Catalog of Multiplex System Considerations (Sheet 1 of 2)

4.0 RELIABILITY ASPECTS

4.1 Design Techniques

4.2 Analytical

4.3 Failure Detection

5.0 ENVIRONMENT

5.1 Power

5.2 Size

5.3 Weight

5.4 Temperature

5.5 Shock

5.6 Vibration

5.7 Acoustical

5.8 Barometric Pressure (Altitude)

5.9 Humidity

5.10 Radiation

5.11 Electromagnetic Interference (EMI)

Figure A-5. Condensed Catalog of Multiplex System Considerations (Sheet 2 of 2) understanding of the problems which each of them overcomes or creates. Usage of these two techniques is generally governed by the following rationale:

- TDM terminals may all be of identical design, which is a definite plus for commonality. However, if one user requires a much higher data rate than all the others, then a special FDM channel may be desirable (e.g., a 10-Mbps channel among a group of 10-kbps channels constrains all channels to a 10-Mbps receive rate, even though they all do not require it).
- TDM terminals work well at baseband, where the transmission line losses are lowest. However, equalization and amplification are more easily achieved in FDM (carrier) systems; therefore, these systems are generally preferred for long transmission paths and very high data rates.

Finally, the command and control section provides a diagramatic summary of the fundamental command and control relationships applicable to information-transfer systems. Although some of the schemes described lend themselves more to one multiplexing scheme than another, no attempt is made to catalog on that basis. A special graphical notation was developed to facilitate description of a wide variety of command and control relationships.

In summary, the efforts of Task Group 1 led to the preparation of a sizable document in which comprehensive and detailed information regarding multiplex system design is presented in well structured form. It clearly illustrates the range of areas in which design decisions must be made, and provides guidelines on the basis of which tradeoffs can be made. However, it does not provide a formal model of the multiplex system design process which indicates, for example, the sequence in which various design decisions should be made. Such a model could serve as the basis for a true "cook book" approach to multiplex system design; but it appears that, at present, development of this type of model is beyond the state-of-the art.

3. TASK GROUP 2 - SIMULATION TECHNOLOGY STUDY

In the case of Task Group 1, the equivalent of a multiplex system designer's handbook was needed as a starting point for the MUXSIM development. Also required is the equivalent of a simulator designer's handbook; and then, given these two items, the definition of MUXSIM could emerge as a result of various screenings and tradeoff studies.

However, whereas the equivalent of a multiplex system designer's handbook had to be created (because an existing one could not be found), the equivalent of a simulator designer's handbook does already exist amidst the voluminous literature on simulation (books, papers, languages, etc.). Therefore, Task Group 2 was easier to perform than Task Group 1, since it consisted mainly of a screening process on a relatively well-defined existing base of simulation technology.

Considering simulation as a whole, it is immediately evident that the range of simulator types is very large. For example, simulators can broadly be classified either by intended usage or by technology, as shown in Figure A-6.

It was assumed during the MUXSIM definition phase that the MUXSIM host system was to be a DEC PDP-11 multiprocessor system as shown in Figure A-7, with possible relationships to multiplex system physical components as shown in Figure A-8. It was further assumed that the desired end product was a simulator which was a cost-effective multiplex system designer's tool, as indicated in Figure A-3. Therefore, given these assumptions, it was tentatively concluded that the desired MUXSIM would probably be of type U2-T1 (see Figure A-6). This conclusion was helpful in restricting the scope of detailed simulation technology investigations.

The principal rationale associated with host-system-related MUXSIM scoping is as follows. First, the host system is generally viewed as a powerful minicomputer system which provides medium-scale computing power at a low price; i.e., at a price sufficiently low that dedicated use of the system for an application such as MUXSIM is economically viable. Second, the small word size of the host system processors, together with their small memories and lower speeds relative to existing larger systems, means that the use of very large models and powerful simulation languages such as GPSS and SIMSCRIPT may be undesirable. Also, simulation experiments which entail very significant amounts of complex number-crunching are best avoided on such a host system. Fianlly, the architecture of the host system is such that "foreign" devices can be attached with relative ease; this is mainly because of the Unibus feature of the system.

In short, the knowledge that MUXSIM had to fit on a minicomputer system was used in the definition phase to restrict the range of alternative simulator types to be considered in detail for MUXSIM. Subsequently, in the implementation phase, the MUXSIM host system was redesignated to be the DEC System-10, which is a large-scale computer system. It is noteworthy that MUXSIM as defined to be suitable for minicomputer system implementation is also suitable for large system implementation; however, the converse wouldn't necessarily be true. The main impact of the original host system assumption was to prevent the definition of MUXSIM as a very large-scale simulator.

With the above in mind, the range of relevant simulation technologies for immediate study was narrowed to non-real-time digital simulation of the continuous or discrete event types, and in forms suitable for implementation on a minicomputer system. Continuous simulation is best suited for describing dynamic systems which are generally characterized by differential equations, while discrete event simulation is best for modeling systems which are characterized by a set of discrete states. In the latter case, the simulation represents system behavior by moving the system from state to state in accordance with well-defined operating rules, and system state changes depend on the occurrence of discrete events; hence the name.

For reasons stated above, it was determined that MUXSIM should probably be a digital simulator useful as a design tool for the multiplex system designer, and suitable for implementation on a minicomputer system. Further focus was required beyond this in

Intended Usage

- U1 - Personnel training (e.g., Link trainers for pilot training, space flight simulators for astronaut training)
- U2 - "What if" question answering (e.g., models for answering questions related to optimal economic policy decisions, models for answering questions related to optimal system design)

Simulator Technology

- T1 - Digital Simulators
- T2 - Analog Simulators
- T3 - Hybrid Simulators (part digital, part analog)

Figure A-6. Simulator Classifications

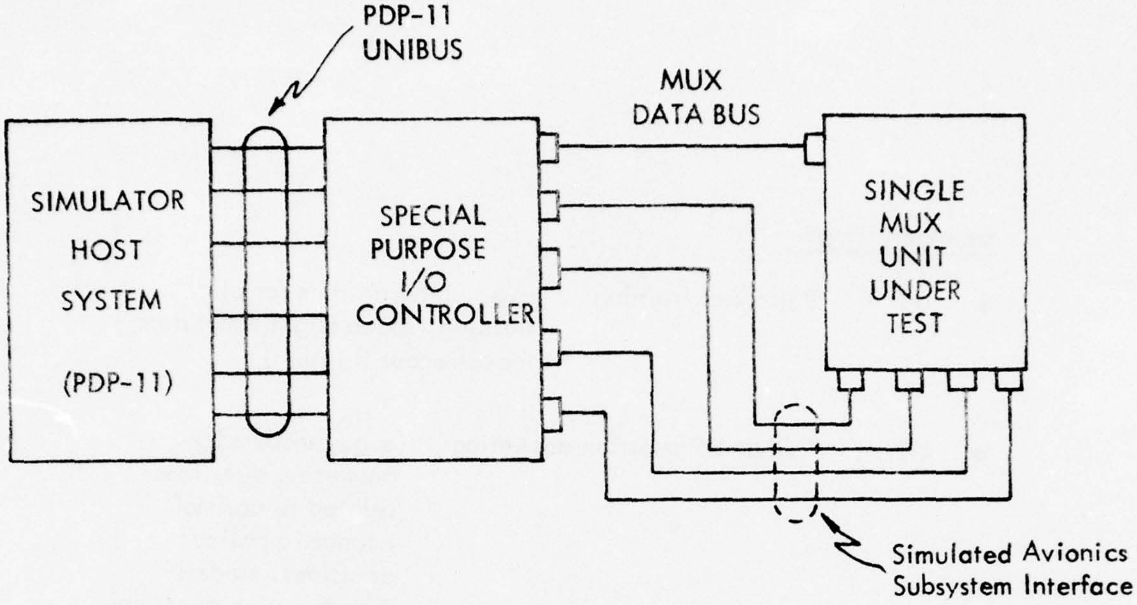


Figure A-7. Typical Test Bed Interface - Single Multiplex System Unit

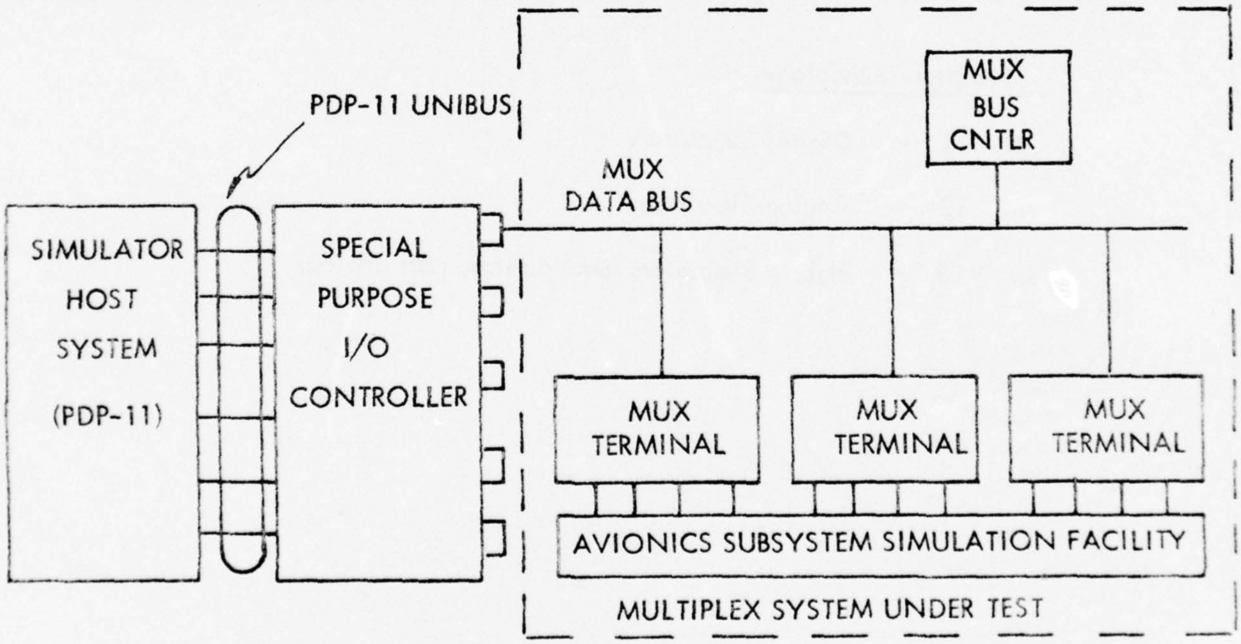


Figure A-8. Typical System Hardware Interface - Complete Multiplex System order to restrict the range of simulation technologies to be considered in detail; and in particular the question of the relation of simulator time to real time had to be considered. In this regard, it was determined that MUXSIM might well have a real-time mode, which could be useful in exercising multiplex system components for example. However, it was also determined that this mode, which was termed the "test bed" mode, should be defined separately from the simulator or non-real time mode, because in the latter the relation between real and simulated time is of no great consequence. Also, it was decided to consider simulation technology for the non-real time mode in detail first.

In summary of the foregoing, in consequence of a logical screening process it was decided to focus initial detailed simulation technology studies in the following areas:

- Non-real time
- Discrete-event
- Suitable for minicomputer implementation

This having been decided, the technology study focused initially on languages suitable for discrete event simulation. This study was conducted using three different approaches, namely: (1) an evaluation matrix comparison of discrete event simulation languages; (2), a benchmark comparison of same; and (3), literature surveys and consultations with external simulation experts on that subject.

Results of the evaluation matrix comparison are shown in Tables A-II(a) and A-II(b). Two tables are required to cover two different host computer systems (the DEC PDP-11 and the DEC System-10). The evaluations matrix is a convenient way to make a systematic comparison of languages while considering a number of features, and to develop a ranking of these languages. However, the approach has its weaknesses, and in this application it is mainly useful for screening purposes. Results of this exercise indicate the tentative conclusion that Fortran and GASP are the top two choices, whether the planned host system is the DEC System-10 or the PDP-11. However, the ranking of other alternative simulation languages is host system dependent (e.g. APL is ranked 8th for the PDP-11, but 4th for the DEC System-10).

In further evaluation efforts, a benchmark approach was used to compare Fortran and GPSS in development of a discrete-event-type model of a simple message handling system. Results of this exercise are shown in Table A-III. Only two languages were compared using the benchmark approach because of the high cost of using the approach (i.e., models must be developed, debugged and run in each language being considered). From this exercise it was concluded that, for models well-suited for the GPSS, the use of GPSS results in substantial reduction of programming effort and a much smaller source program, but it entails the use of much greater computer resources for program execution. At the time this exercise was done, it was still assumed that the MUXSIM host computer system was to be the PDP-11; and this exercise indicated that GPSS was not a good simulation language for use with the PDP-11.

Table A-II (a). MUXSIM IMPLEMENTATION LANGUAGE EVALUATION MATRIX (PDP-11)

	LANGUAGE PROPERTY → CANDIDATE LANGUAGE ↓	① Availability for PDP 11	② Suitability for PDP 11 configuration	③ Logical suitability for MUX SIM	④ Efficiency (speed) of object code	⑤ Efficiency (core size) of object code	⑥ Availability of knowledgable programmers	⑦ Ease of learning for new user	⑧ Ease of use for programming MUX SIM (no. of lines of source code)	⑨ Ease of modification or extension	⑩ Ease of use with non-standard peripherals	⑪ Availability on other computers	⑫ Viability (likelihood of continuing usage)	⑬ Suitability for modular programming	⑭ Suitability for MUX SIM in general	Total Score	% Score	Rank
	1. ASSEMBLY	10	10	10	10	10	7	3	2	10	10	1	8	10	10	1750	83.7	3
General Purpose	2. FORTRAN	10	10	8	8	8	9	6	6	9	9	9	10	10	9	2030	97.2	1
	3. ALGOL - like	10	10	8	8	8	6	6	5	9	9	6	8	10	9	1726	82.6	4
	4. COBOL	0	5	5	6	6	9	6	4	6	6	9	10	6	6	1185	56.7	9
	5. PL/1	0	1	9	7	6	8	5	7	9	9	6	9	10	5	1334	63.8	5
	6. APL	0	1	9	5	9	6	5	9	10	5	8	7	10	5	1243	59.5	8
Simulation	7. GASP-like	9	10	8	8	8	4	6	9	9	9	9	8	10	9	1815	86.8	2
	8. GPSS-like	0	5	7	5	6	6	6	7	7	6	6	8	9	7	1280	61.2	7
	9. SIMSCRIPT-like	0	1	9	8	8	6	4	8	9	9	4	8	10	6	1326	63.4	6
Special Purpose	10. CDL	0	5	7	7	7	4	5	7	5	6	4	6	9	6	1167	55.8	10
	11. ISP/PMS	0	5	8	7	7	2	4	7	5	6	1	5	9	6	1093	52.3	12
	12. SCERT	0	5	7	8	7	2	4	6	8	6	1	6	10	6	1151	55.1	11
	PROPERTY WEIGHTS	20	20	20	5	8	8	5	20	15	15	15	18	20	20	☒	☒	☒

NOTES:

1 Properties are selected as "good" attributes. Thus, each language can be assigned a score for each property (say, from 1 to 10, with 10 best).

2 Score for each language can be obtained as a weighted sum of property scores, where weights for the various properties are obtained by consensus of knowledgeable individuals.

Table A-II (b). MUXSIM IMPLEMENTATION LANGUAGE EVALUATION MATRIX (DEC SYSTEM-10)

	LANGUAGE PROPERTY → CANDIDATE LANGUAGE ↓	① Availability for DEC-10	② Suitability for DEC-10 configuration	③ Logical suitability for MUX SIM	④ Efficiency (speed) of object code	⑤ Efficiency (core size) of object code	⑥ Availability of knowledgable programmers	⑦ Ease of learning for new user	⑧ Ease of use for programming MUX SIM (no. of lines of source code)	⑨ Ease of modification or extension	⑩ Ease of use with non-standard peripherals	⑪ Availability on other computers	⑫ Viability (likelihood of continuing usage)	⑬ Suitability for modular programming	⑭ Suitability for MUX SIM in general	Total Score	% Score	Rank
General Purpose	1. ASSEMBLY	10	10	10	10	10	7	3	2	10	10	1	8	10	10	1700	81.3	5
	2. FORTRAN	10	10	8	8	8	9	6	6	9	9	9	10	10	9	1851	88.6	1
	3. ALGOL - like	10	10	8	8	8	6	6	5	9	9	6	8	10	9	1726	82.6	3
	4. COBOL	10	10	5	6	6	9	6	4	6	6	9	10	6	6	1495	71.5	8
	5. PL/1	0	10	9	7	6	8	5	7	9	9	6	9	10	5	1514	72.4	6
	6. APL	10	10	9	5	9	6	5	9	10	5	8	7	10	5	1701	81.4	4
Simulation	7. GASP-like	10	10	8	8	8	4	6	9	9	9	9	8	10	9	1835	87.8	2
	8. GPSS-like	0	10	7	5	6	6	6	7	7	6	6	8	9	7	1380	66.0	10
	9. SIMSCRIPT-like	0	10	9	8	8	6	4	8	9	9	4	8	10	6	1506	72.1	7
Special Purpose	10. CDL	10	10	7	7	7	4	5	7	5	6	4	6	9	6	1461	69.9	9
	11. ISP/PMS	0	10	8	7	7	2	4	7	5	6	1	5	9	6	1197	57.3	12
	12. SCERT	0	10	7	8	7	2	4	6	8	6	1	6	10	6	1245	59.6	11
	PROPERTY WEIGHTS	20	20	20	5	8	8	5	20	15	15	15	18	20	20	☒	☒	☒

NOTES:

1 Properties are selected as "good" attributes. Thus, each language can be assigned a score for each property (say, from 1 to 10, with 10 best).

2 Score for each language can be obtained as a weighted sum of property scores, where weights for the various properties are obtained by consensus of knowledgeable individuals.

Table A-III

FORTRAN VS. GPSS FOR MESSAGE-HANDLING SYSTEM BENCHMARK

	FORTRAN	GPSS
1. Program Name	MHS 1	MHS 3
2. Programming Effort (Man-Days)	2	1
3. Debugging Effort (Man-Days)	1	1
4. Size of Source Program (No. of Lines)	330	28
5. Computer Resource Units (CRU) for computation and execution	3	24

Finally, the literature survey/expert consultation approach toward simulation language evaluation produced several published simulation language inventories and comparisons, and revealed a number of interesting details. The main result of the effort was to strengthen the impression that neither GPSS nor SIMSCRIPT was a good choice of implementation language for MUXSIM, *given that the PDP-11 was to be its host system;* but that a Fortran-based discrete event simulation language called GASP was in existence, well documented, well-supported by an independent software supplier, and well-suited for implementation in a minicomputer system. In a crude estimate of simulation language "power", GASP was found to occupy a position somewhere between GPSS and SIMSCRIPT (the latter being one of the most powerful discrete event simulation languages available). In addition to discrete event simulation languages, several mature and successful simulation packages such as SCERT (System and Computer Evaluation and Review Technique) were examined in some detail as part of the literature survey/expert consultation approach, but none were found to be directly applicable to MUXSIM.

In summary, the efforts of Task Group 2 entailed a broad survey of the entire field of simulation technology with particular emphasis on discrete event simulation languages that were both mature and widely used. From this effort it was tentatively concluded that MUXSIM should be implemented in Fortran and/or GASP, and should incorporate some of the good features found in mature and successful simulation packages such as SCERT. Areas of simulation technology such as real-time simulation techniques, analog and hybrid simulation, and continuous simulation languages were not examined in detail pending demonstration of the need for such investigations. In hindsight, it is noteworthy that portions of MUXSIM have been implemented in a recent version of GASP called GASP IV, which has both discrete event and continuous simulation capabilities. However, the latter have not yet been used in models implemented to date.

4. TASK GROUP 3 - IDENTIFY SIGNIFICANT MUX DESIGN QUESTIONS

The purpose of this task group was to identify those significant multiplex system design questions which are best answered by simulation. In general, a design question is regarded as significant if the cost-effectiveness of the resultant multiplex system is strongly affected by its answer; and the question is best answered by simulation if its answer cannot be arrived at better by some other means available to the multiplex system designer.

This identification was made mainly by a judgmental process based on experience with multiplex system design and simulation technology generally, and on the results of Task Groups 1 and 2, specifically. The process is explicitly displayed as a three-phase screening process in Figure A-9. A major assumption leading to the results of this effort was that, to ensure maximum benefits to the Air Force from MUXSIM use, MUXSIM should be applicable at the earliest possible time in the multiplex system design cycle. Rationale for this assumption is that early MUXSIM applications have the greatest leverage and the least costs. It is true that simulation can be done at almost any stage of the design cycle, and in fact a hardware prototype can be viewed as a special case of a detailed design simulation. However, the later in the design cycle that simulation is used, the greater the cost of simulation model development and the greater the cost of design changes indicated by simulation results. If design errors are found too late in a design cycle, a major design change may be impractical for cost and schedule reasons, and thus a "band-aid" type fix may be necessary.

Because of this assumption, the focus of MUXSIM was narrowed so as to make it applicable mainly to those significant design questions which must be answered earliest. These are the "big picture" or "systems" type questions, the answers to which determine the basic multiplex system architecture. Examples are:

- What is the required bus speed?
- What is the bit error rate on the bus?
- What is the optimal bus command and control scheme?
- How many times per second should a given bus signal be sampled?
- How many terminals will be needed?
- What is the maximum acceptable per unit cost for the multiplex system?
- Should bus control be centralized or distributed?
- What is the required redundancy, and how should it be managed (what is the required multiplex system MTBF)?
- What is the impact of varying multiplex system MTBF on life cycle costs of a fighter aircraft?

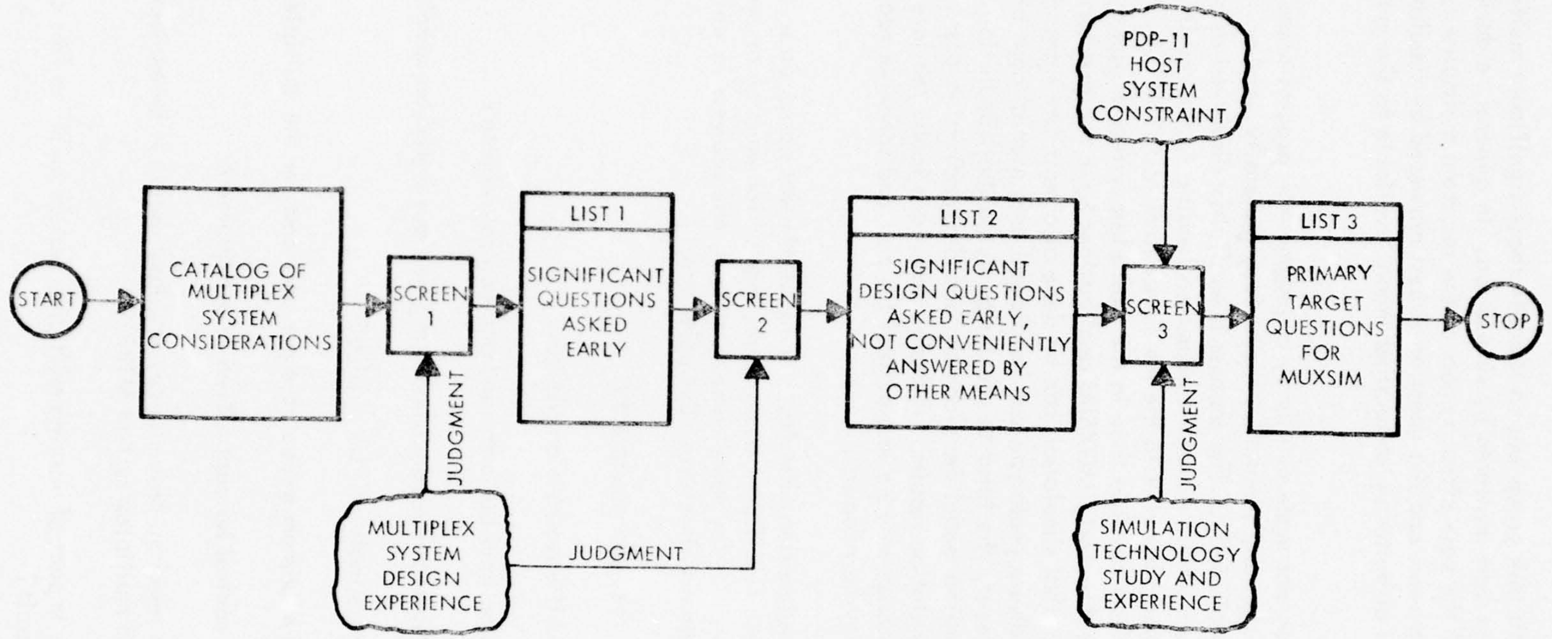

START
CATALOG OF MULTIPLEX SYSTEM CONSIDERATIONS
SCREEN 1
JUDGMENT
MULTIPLEX SYSTEM DESIGN EXPERIENCE
LIST 1
SIGNIFICANT QUESTIONS ASKED EARLY
JUDGMENT
SCREEN 2
LIST 2
SIGNIFICANT DESIGN QUESTIONS ASKED EARLY, NOT CONVENIENTLY ANSWERED BY OTHER MEANS
PDP-11 HOST SYSTEM CONSTRAINT
SCREEN 3
JUDGMENT
SIMULATION TECHNOLOGY STUDY AND EXPERIENCE
LIST 3
PRIMARY TARGET QUESTIONS FOR MUXSIM
STOP

Figure A-9.

Identification of Primary Target Questions for MUXSIM

- What is the probable reliability of a candidate multiplex system architecture?

- What is the probable cost of implementing a particular multiplex system design?

- What is the flexibility of a particular multiplex system architecture (how readily can new signals be added, etc.)?

- Should TDM or FDM be employed?

Questions like these comprise List 1 in Figure A-9.

Although the above questions are all significant and asked early, not all are best answered by simulation (and specifically by simulation on a minicomputer system). For example, several areas of multiplex system design are best handled on an analytic basis. They include bus-related matters such as bit error rate, impedance matching and transmission line problems, TDM vs FDM tradeoff studies, etc. A general rule used in screening questions was that, if a cost-effective alternative means existed for answering a given design question adequately, then that question should not be considered a prime target for MUXSIM. After this second screening, the questions that remain are those of List 2 in Figure A-9. Typical examples are:

1. What is the optimal bus command and control scheme?

2. What is the bus loading for a given bus command and control scheme?

3. What is the specific assignment of signals to bits, words and messages for a given bus command and control scheme and a given signal flow list?

4. What redundancy is needed, and how is it best managed?

5. What is the best sequence for placing information on the bus for transfer?

6. How many terminals are needed, and how should signals be allocated to terminals?

7. Should processing capacity be placed in the terminals, and if so how much?

8. What is the impact of varying multiplex system MTBF on life cycle costs of a fighter aircraft.

From this it is evident that List 2 can contain questions (e.g., question 8) which are significant and well-answered by simulation, but for which the size or complexity of the implied models is very great. At this point it was recalled that the MUXSIM host system is a minicomputer system with limited memory size and processing "horsepower", and questions inconsistent with these limitations were deleted. This step constituted Phase three of the three-phase question screening process, and List 3 is the resultant list of questions. For economy of description, it is best characterized by the three screening steps mentioned above, rather than by an explicit question list.

5. TASK GROUP 4 - IDENTIFY MUXSIM FUNCTIONAL REQUIREMENTS

Although the principal functional requirements of MUXSIM are largely determined (albeit indirectly) by the list of primary target design questions, it was necessary to proceed further and give explicit requirements for MUXSIM inputs, outputs, variables of simulation, control mechanisms, etc. These matters are covered in this section, which presents an integrated functional conceptual-level description of MUXSIM as a whole. This is the major result of the Task Group 4 effort. The description is that of a fairly large and comprehensive simulator, but it is not implied that the first version of MUXSIM built should implement all of it. It is primarily provided by Figures A-10 and A-11, together with supporting text and other associated figures. Figure A-10 provides a very general view of the role of MUXSIM in the avionics multiplex system design process, while a more detailed functional view is provided in Figure A-11. These are the key figures in this system, and they are referred to often in the discussions that follow.

a. MUXSIM Inputs (including Variables of Simulation)

Under the heading of Inputs, the following subjects are covered in the indicated sections:

(1) System and Workload Description Inputs
- (a) Signal Flow Listing (Level 2)
- (b) Avionics Systems Listing (Level 3)
- (c) Air Vehicle Selection (Level 4)
- (d) Simulator Inputs (Level 1)
- (e) "Prediction" and "Growth" Inputs
- (f) Canned Inputs

(2) Simulator Control Inputs

(3) Test Bed Inputs

All these inputs are shown in Figures A-10 and A-11, with the exception of test bed inputs which are treated separately.

Consider first Figure A-10, and recall that the basic "real world" problem that MUXSIM addresses is that of information transfer. Often the basic problem concerns some futuristic requirement which is not accurately defined, and which implies requirements for growth and flexibility which are subject to judgment or prediction. The information characterizing and influencing the information transfer system requirements is that referred to as "General Inputs" in Figure A-10. That information which describes the candidate information transfer system which is supposed to meet the information transfer requirements is referred to as the "Variables of Simulation" in Figure A-10. That information which tells the multiplex system designer what he really wants to know, e.g.,

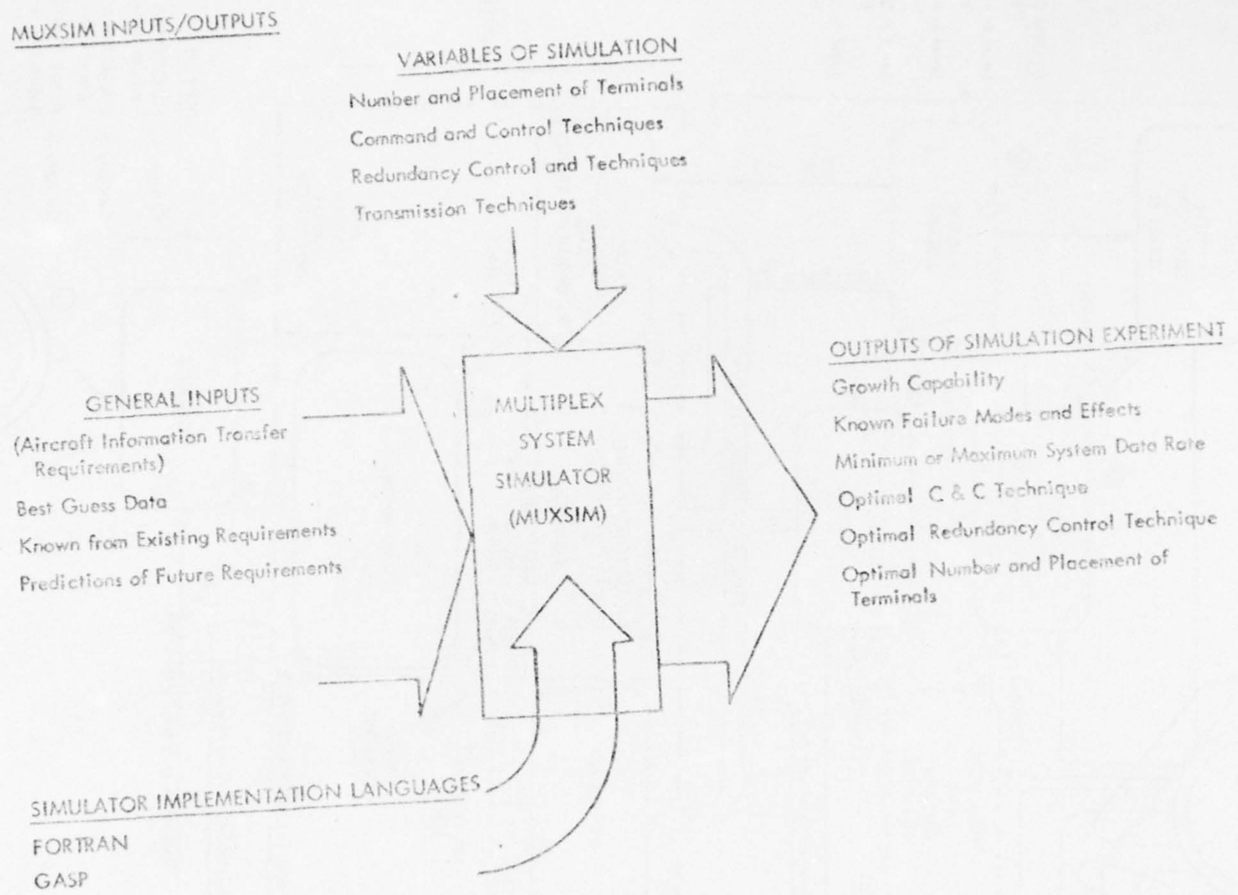

MUXSIM INPUTS/OUTPUTS
VARIABLES OF SIMULATION
Number and Placement of Terminals
Command and Control Techniques
Redundancy Control and Techniques
Transmission Techniques
GENERAL INPUTS
(Aircraft Information Transfer Requirements)
Best Guess Data
Known from Existing Requirements
Predictions of Future Requirements
MULTIPLEX SYSTEM SIMULATOR (MUXSIM)
OUTPUTS OF SIMULATION EXPERIMENT
Growth Capability
Known Failure Modes and Effects
Minimum or Maximum System Data Rate
Optimal C & C Technique
Optimal Redundancy Control Technique
Optimal Number and Placement of Terminals
SIMULATOR IMPLEMENTATION LANGUAGES
FORTRAN
GASP

Figure A-10.
MUXSIM Inputs and Outputs

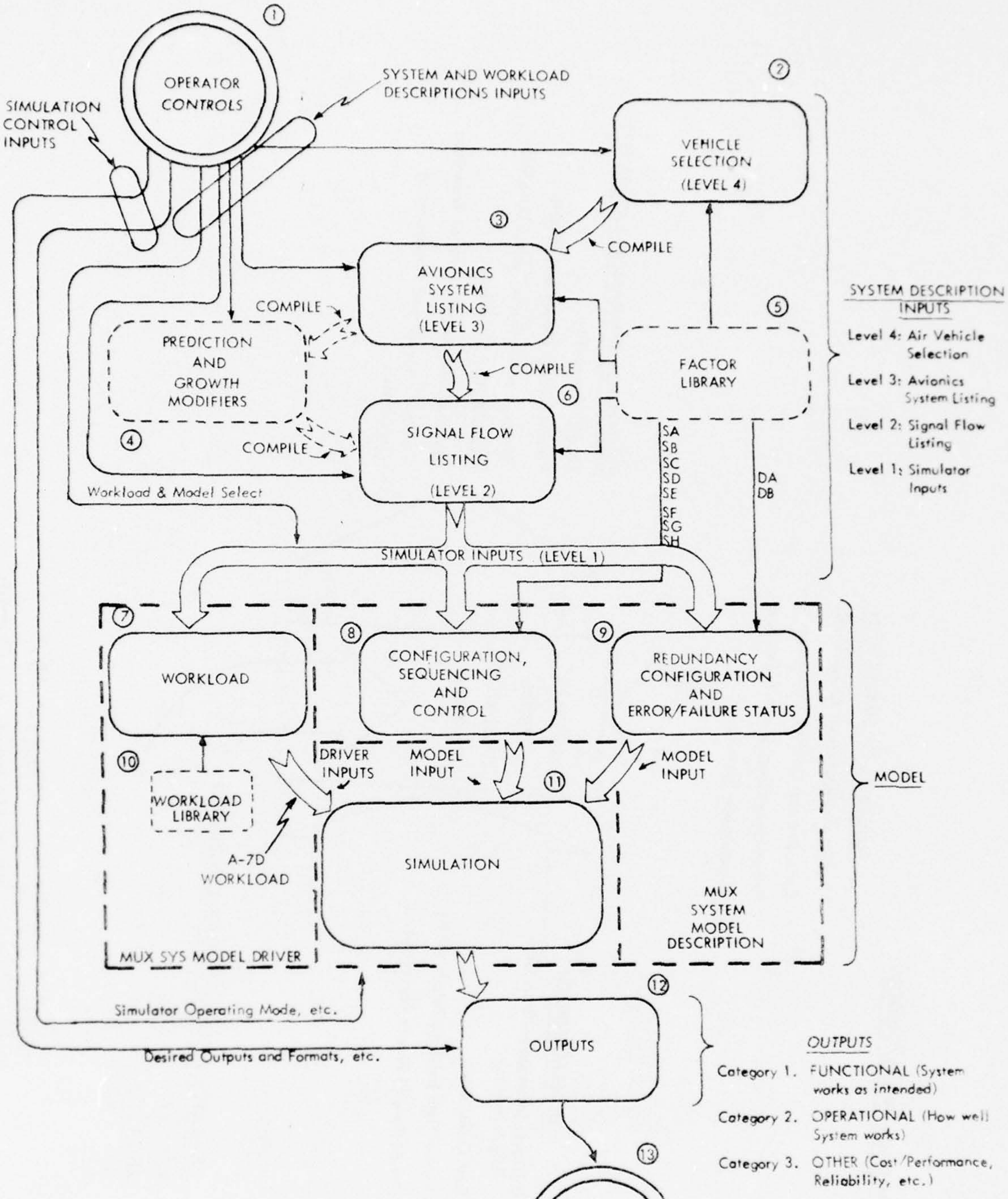

1
OPERATOR CONTROLS
SIMULATION CONTROL INPUTS
SYSTEM AND WORKLOAD DESCRIPTIONS INPUTS
2
VEHICLE SELECTION (LEVEL 4)
COMPILE
3
AVIONICS SYSTEM LISTING (LEVEL 3)
COMPILE
PREDICTION AND GROWTH MODIFIERS
4
COMPILE
COMPILE
5
FACTOR LIBRARY
6
SIGNAL FLOW LISTING (LEVEL 2)
SA
SB
SC
SD
SE
SF
SG
SH
DA
DB
Workload & Model Select
SIMULATOR INPUTS (LEVEL 1)
SYSTEM DESCRIPTION INPUTS
Level 4: Air Vehicle Selection
Level 3: Avionics System Listing
Level 2: Signal Flow Listing
Level 1: Simulator Inputs
7
WORKLOAD
8
CONFIGURATION, SEQUENCING AND CONTROL
9
REDUNDANCY CONFIGURATION AND ERROR/FAILURE STATUS
10
WORKLOAD LIBRARY
DRIVER INPUTS
MODEL INPUT
11
MODEL INPUT
MODEL
A-7D WORKLOAD
SIMULATION
MUX SYSTEM MODEL DESCRIPTION
MUX SYS MODEL DRIVER
12
Simulator Operating Mode, etc.
OUTPUTS
Desired Outputs and Formats, etc.
OUTPUTS
Category 1. FUNCTIONAL (System works as intended)
Category 2. OPERATIONAL (How well System works)
Category 3. OTHER (Cost/Performance, Reliability, etc.)
13
OPERATOR PRESENTATION

Figure A-11. MUXSIM System - Conceptual Level which of several alternative design schemes or techniques best meets the requirements, is referred to as "Outputs of the Simulation Experiment." It should be noted that such information is not generally the result of a single run or exercising of a single model, but rather is the result of a number of runs on a multiplicity of models, which in the aggregate may be referred to as a "Simulation Experiment," and which has a different meaning from a simple simulation. Those languages which assist in the translation from a formal description of the multiplex system and its workload to a working software model of the multiplex system, together with the data for driving the model, are referred to as "Simulator Implementation Languages." They are usually either general purpose programming languages or general purpose simulation languages.

Next consider Figure A-11. It presents a somewhat different but equivalent view of MUXSIM inputs and outputs. It provides a more detailed and analytic view of the situation, and more directly supports the discussions in this section. Therefore it should be regarded as the primary illustration for points made hereinafter.

(1) System and Workload Description Inputs

In order to use MUXSIM in his system design efforts, the multiplex system designer must ultimately provide precise formal descriptions of both the alternative system models he wishes to study and the workloads he wishes to apply to those models. However, in the course of arriving at this level of description, which is referred to as Level 1 of the inputs in Figure A-11, he may first work with higher level and more indirect descriptions of the information transfer problem from which he subsequently derives the Level 1 inputs; and this derivation may be by manual, semiautomatic, or automatic means. These higher level descriptions are referred to as Levels 2, 3, and 4 in Figure A-11, and in general the higher the level of the input, the more indirect the description of the information transfer system that it embodies. In fact, a Level 4 description may indirectly describe the null multiplex system, for it may describe a type of air vehicle for which a multiplex system is not at all feasible. The entire set of hierarchical information input levels is essentially defined as an effort to structure the multiplex system design process in such a way as to facilitate the application of computer-aided design techniques at different stages in this process. The blocks shown in Figure A-11 are discussed below on an individual basis.

(a) Signal Flow Listing (Level 2)

This level of information inputs is discussed first because it is the primary source of information transfer requirements for the multiplex system, from which the Level 1 inputs must subsequently be derived. The signal flow listing consists of the individual signals which carry the information among the systems within the aircraft. The avionics signal types from which the signal flow listing is derived are shown in Table A-IV. The signal flow listing itself contains, for each signal, all pertinent information required to specify the signal characteristics which are relevant to the multiplex system design problem. It will be available initially in the form of a card deck and will contain the information shown in Table A-V.

Table A-IV. AVIONICS SIGNAL TYPES

Discrete (single bi-level channel):

Information is in the signal position.

Analog (single continuous channel):

Information is in the time duration of the signal.
Information is in the amplitude of the signal.
Information is in the amplitude of the modulated signal.

Resolver (dual continuous channels):

Information is in the amplitude of the two modulated signals.

Synchro (triple continuous channels):

Information is in the amplitude of the three modulated signals.

Parallel Digital (multi bi-level channels/time dependent):

Information is in the signal position at each channel during a given time interval.

Serial Digital (single bi-level channel/time dependent):

Information is in the signal position during a sequential series of time intervals.

Pulse (single channel/time dependent):

Information is contained in the time that the pulse occurs.

Pulse Rate (single channel/time dependent):

Information is contained in the number of pulses per unit time.

Table A-V. SIGNAL FLOW LISTING INFORMATION

Signal Name

Name *assigned to identify the signal.*

Mission Phase

Identification, by listing, of the various phases of the mission where this signal might be used.

Mission Function

Identification, by listing, of the major avionic functions of which this signal is a part (i.e., navigation, communication, etc.).

Origin Location

Identification, by alphanumerics, of the aircraft X, Y, and Z axes of the origin LRU.

Destination

Identification, by listing, of the LRU where the signal will *terminate.*

Destination Location

Identification, by alphanumerics, of the aircraft X, Y, and Z axes of the destination LRU.

Equation Number

Identification, by alphanumerics, for signals which originate or are *delivered to a* processor (the number of the equation in the computations task description). *(Multiple usage will result in multiple listings under this heading.)*

Multiplicity Number

Identification, by numerics, of the total number of destinations for multi-destination signals. The signal listing will be repeated for each usage with the same signal name and origin, *but with a different destination.*

Signal Type

Identification, by listing, of the general classification of the signal (i.e., discrete, dc, analog, synchro, etc.).

Frequency Content

Identification (where applicable), by numerics, of the highest frequency in Hz contained in the information portion of the signal; that is, the highest frequency contained in the equivalent baseband transmission of the signal.

Update Rate

Identification, by numerics, of the number of times per second that the signal would have to be transmitted or updated in a Time Division Multiplex (TDM) *system to satisfy the user (destination) requirements.* Note that this is a function of particular application and output reconstruction requirements.

Quantization (Resolution) Bits

Identification, by numerics, of the number of signal bits that the particular signal would require to pass the information to the user.

Delay Sensitivity

Identification (where required), by numerics, of the maximum delay in milliseconds which the signal could tolerate.

Skew Sensitivity

Identification (where required), by numerics, of the maximum skew allowable as a function of percent of the update period.

(b) Avionics Systems Listing (Level 3)

Similar types of equipment, or subsystems, on board aircraft generally have the same data requirements. Moreover, these subsystems usually have their equipment distributed within the same physical areas of their host aircraft (i.e., the forward-looking radar, flight control system, cockpit and its associated systems, etc.). A useful input baseline can therefore be compiled by specifying the avionic systems contained in the vehicle.

(c) Air Vehicle Selection (Level 4)

Specific classes of aircraft have similar mission requirements and consequently similar avionics hardware or system requirements. Therefore, an input baseline can be specified from which an effective signal flow listing can be compiled by simply specifying the aircraft type. Table A-VI shows a classification of aircraft types which is considered suitable for this application.

(d) Simulator Inputs (Level 1)

As Figure A-11 shows, there are three different types of Level 1 inputs: "workload," "sequencing and control," and "configuration and error/failure status." The latter two types comprise the information referred to in Figure A-10 as "Variables of Simulation," while the former may be distilled from portions of the information referred to in Figure A-10 as "General Inputs." The MUXSIM user must specify inputs at Level 1 formally and precisely, because the MUXSIM system itself must translate from these into object code which constitutes the multiplex system model and its workload; therefore, ambiguity cannot be tolerated. Our present discussion will cover the contents of the information at this level, but not its format, since specification of the latter has not yet been done. Format specification is one of the primary tasks remaining in this study.

There is an intimate relationship between Level 2 inputs and Level 1 inputs because the former represents a rather precise description of the information transfer requirements for which Level 1 inputs are intended to be a system design solution. From these requirements can be derived, by various means, the description of both candidate multiplex systems and "typical" workloads to be executed by such systems. Figure A-11 is intended to show that portions of this derivation can be performed by automatic means, i.e., with the aid of digital computer programs. Such programs can be referred to as "simulator input generator" routines. Representative tasks for these routines are: 1) the task of making optimal assignments of signals to remote terminals; and 2) the task of deriving information traffic statistics which are typical of anticipated real world traffic situations for the multiplex system. The latter task is of great importance, because even a valid model of a multiplex system cannot be used to fully demonstrate adequacy of the design unless the model is driven by a realistic workload.

At Level 1, "configuration" inputs are those which declare what the component subsystems of a multiplex system are, while "sequencing and control" inputs are those which describe the interconnections and interactions of these subsystems

Table A-VI. CLASSIFICATION OF AIRCRAFT TYPES

● Close Air Support	Day Night Night/All weather
● Air Superiority	Day Night Night/All weather
● Interceptor	All Weather
● Bomber/Strategic	Light Medium Heavy
● Photo Reconnaissance	
● Surveillance	Infrared Electronic
● Gunship	
● Countermeasures	

during actual system operation. In MUXSIM, these subsystems and their interactions are implemented by generalized functions and by the control program which sequences these functions and provides communications between them. "Error/failure status" inputs are a special type of configuration inputs which describe the "health" of the component subsystems. Thus, the MUXSIM user may declare that a certain subsystem has failed or is producing intermittent errors, and by running his simulation model in this condition he may check for adequate redundancy management behavior and for adequate performance in the presence of errors.

"Workload" inputs are those that describe the input information flow which it is the job of the multiplex system to manage. Thus, the workload inputs describe input information traffic scenarios which may vary widely in sophistication from a simplistic workload which generally checks for minimal system functioning, to a complex workload which portrays varying information traffic loads during several phases of a complete aircraft mission. These inputs, like the other Level 1 inputs, must be supplied precisely and formally in accordance with a specific format which has not yet been developed.

(e) "Prediction" and "Growth" Inputs

As previously discussed, a system is often studied (by the use of the simulator) in the context of some future application for which the exact requirements are usually unknown. In such cases, the only factors that might be definable are the aircraft type and possibly the equipment types that are to be used in the aircraft. Level 3 and Level 4 inputs are intended to be of use here, and they have to be modified to accommodate anticipated aircraft systems growth and expansion requirements. Although the designer's crystal ball is murky, he may have some feel for anticipated future developments and for trend predictions. These are discussed next.

Anticipated Future Developments

Usually the best information available is that for present equipment. One forecasting approach is to modify such information through engineering judgment on a signal-by-signal basis. Among the points that need to be considered are future developments resulting from programs concerning Integrated Controls and Displays, Digital Flight Control Systems, and other such systems. For example, Integrated Controls and Display Systems reduce the number of interfaces in the cockpit, while other programs such as Standard Interface will reduce the number of different interfaces between equipments. In fact, Standard Interface may eventually result in a single serial-digital interface. Other programs promoting Integrated Digital Avionics may reduce the overall equipment communications requirements by making more efficient use of information. With knowledge of which programs are likely to impact the multiplex system design problem, the user can create a Signal Flow Listing which has been manually modified to incorporate such anticipated changes.

Trend Predictions

A second forecasting approach involves the creation of trend predictor curves which can be applied in a statistical manner to modify the Signal Flow Listing. A program could be written to make these changes automatically. An automatic method like this would typically involve user-selected trend predictors. For example, a user interested in general information might need a trend predictor method that would work by simply keying in the future year of interest, together with other inputs used in calling up canned data. Trend predictors would be based on factors such as Integrated Avionics, Integrated Controls and Displays, etc. Some typical predictor curves are shown in Figure A-12. Even a Level 1 user who specifies simulator inputs directly can use trend predictor techniques to modify his original data, and thereby check the adequacy of his results in the light of various future possibilities.

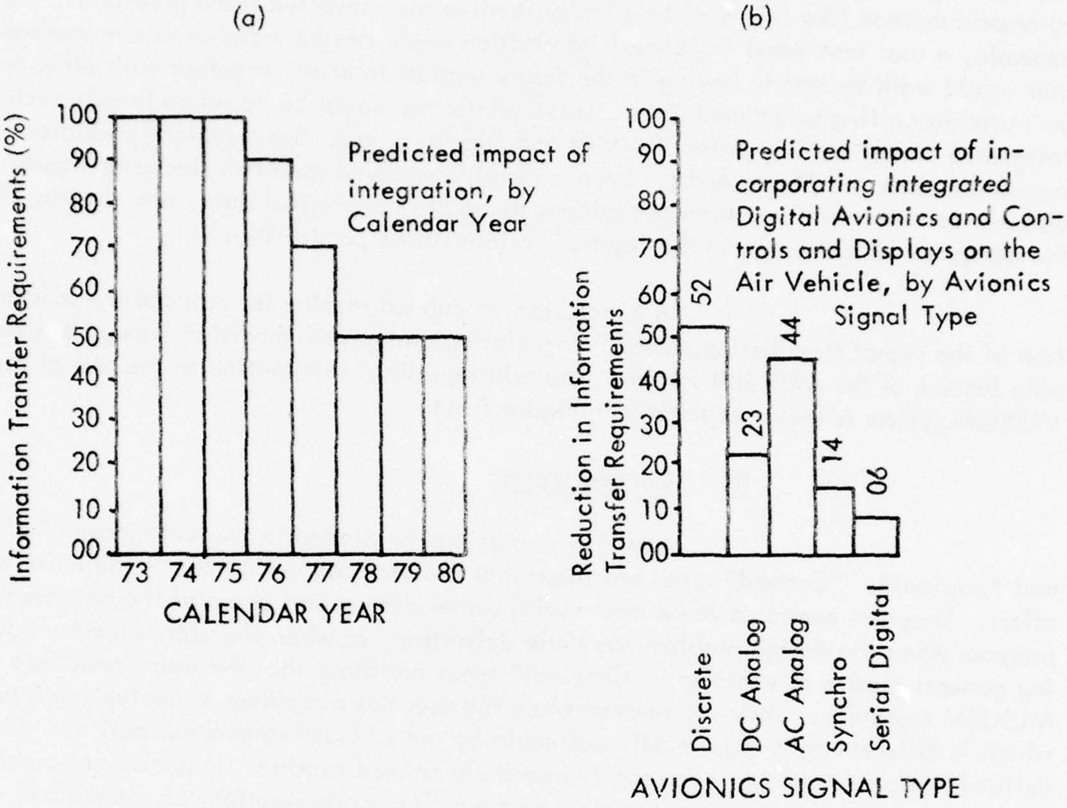
In conclusion, a canned routine for automatic modification of the signal flow listing,called a "prediction and growth modifier," may be a desirable feature of the MUXSIM system. The relationship of this routine to the rest of the MUXSIM system is shown explicitly in Figure A-11.

(f) Canned Inputs

MUXSIM inputs can be placed in two categories: "canned" and "original". "Canned" types are those that the user can define with a minimum of effort. They are expected to be most useful either during that phase of the development program when the design problem has little definition, or when the user is merely addressing general classes of vehicles. "Original" types are those that the user enters into the MUXSIM manually. They are needed when the user has a problem of evaluating a system which is different from those easily definable by use of existing canned routines. Once defined, an original input can readily become a canned routine. It is also anticipated that there will exist a requirement for inputs which are combinations of canned and original types. Several sets of canned inputs are to be resident within the MUXSIM system. In Figure A-11 these canned routines comprise the blocks referred to as "Prediction and Growth Modifiers," "Factor Library," and "Workload Library." Their relationship to the remainder of MUXSIM is indicated in the figure. At present, these blocks are included in the illustration as a conceptual aid, and neither the exact contents nor the formats of these canned sets are presently well-defined. Many factors are relevant to decisions concerning their exact nature and form. Among the most important factors are MUXSIM implementation cost and schedule guidelines.

(2) Simulator Control Inputs

Simulator control inputs are those inputs which provide information about the desired simulator mode of execution, and about the desired selection and presentation of simulator outputs. For example, in the debug phase of a MUXSIM model development it may be desirable to have the simulator run for short intervals in the trace mode, using certain simple standard workload inputs for which outputs are known. In later phases, longer runs with more complex workloads and hard-copy outputs may be desired. Simulator control inputs will allow the MUXSIM user to select these modes of operation according to his needs. Details of these inputs remain to be developed.



(a) Figure shows how the total information transfer requirements for a fixed set of avionics functionsmight decrease with time. Such a reduction could be due to a decrease in the amount of redundant information transferred in an integrated avionics system.

(b) Figure shows the reduction in information transfer requirements that might be achieved in future for a fixed set of avionics functions. This reduction would be due to use of integrated avionics systems, and is shown by avionics signal type.

Figure A-12. Typical Trend Predictor Curves

(3) Test Bed Inputs

The only anticipated requirement that the MUXSIM System will have for real-time interface with hardware is that of a hardware exerciser, or test bed. Inputs associated with this function are not shown in Figure A-11. Because of the associated real-time requirements, the simulator will probably not be able to run elaborate programs in this mode of operation; instead, simulator-to-hardware communications will probably be handled on a table-lookup basis.

These communications will be conducted through a special-purpose I/O control unit. Outputs from the simulator to the hardware will be read out sequentially upon request from the I/O control unit. Inputs from the hardware being tested will simply be brought in from the I/O control unit and compared with information contained in simulator tables. Either a single multiplex system unit or an entire multiplex system can be tested in this manner (see Figure A-8). In either case, the simulator inputs will be provided by loading into the simulator a table containing all the detailed information required by the hardware being tested. These inputs will be pre-prepared, either manually or by processing of information obtained during the running of other non-real-time simulations.

b. MUXSIM Outputs

The identification of MUXSIM outputs was the most difficult task in the MUXSIM definition phase. One way to arrive at this identification indirectly is to view MUXSIM as a multiplex system analysis tool, and to determine the set of designer's questions which the tool should be most helpful in answering. This approach has been discussed in paragraph 4. That material will not be covered again here, but because of the crucial importance of the MUXSIM outputs,an equivalent rationale for their determination is presented below. The context for the discussion is the overall multiplex system design process.

The requirements for a multiplexed Information Transfer System are a prime consideration throughout the conceptual design stages of any military air vehicle. These requirements, when combined with the air vehicle subsystems requirements, yield major design decisions concerning the level of sophistication of the information transfer system. The resultant system may be as complex as a multiple-bus system, which uses a separate information transfer system for each major air vehicle function, such as avionics, weapon stores management, electrical power distribution and performance monitoring. At the extreme other end of the selection scale is an information transfer system which does not use multiplexing at all.

The multiplex system design process is both interactive and iterative, and varies widely in complexity. For instance, if the air vehicle equipments are designed to meet a Standard Interface Requirement, then a multiplexed information transfer system may be easily implemented. Likewise, the process is simplified if there is a commonality of design between the air vehicle subsystems which would allow them to use a common information transfer system. In any event, the decision made during the preliminary design phase will be one of the following three:

- Select multiple multiplexed information transfer systems, with each system serving a major A/V partition (multi-bus system).

- Select a single multiplexed information transfer system which serves one or more A/V partitions (single-bus system).

- Select a non-multiplexed system (no bus system).

MUXSIM must provide insight and guidance at this point in the design phase, and yet not consume vast amounts of time and effort in a complex, total system evaluation. To facilitate attainment of this goal we define three categories of outputs: Functional, Operational, and Other. These are discussed next.

(1) Functional Outputs

The functional outputs of MUXSIM are those which tell the user if the system he has hypothesized will work correctly and with at least adequate nominal and error/fault performance. And, if it does not work properly, the functional outputs will tend to indicate why. As an example, one may consider a system with message priority where high priority messages in the first terminal accessed can prevent access to the remaining terminals in the system, regardless of the priority of their messages. This problem is caused by a conceptual design error, and a detailed simulation of this system is unwarranted.

The need for functional outputs grows as information transfer systems grow in complexity by the addition of features such as priority, store and forward, multiple channel with channel selection, and sophisticated redundancy management schemes. These features add so many complex logical interactions to the system that simulation becomes the only practical way to determine their effects.

Simulation languages such as GPSS, GASP, and SIMSCRIPT offer means for easily obtaining functional outputs. This is particularly true when the system model is operated step by step to observe its detailed behavior when errors are introduced to the various system elements. This type of functional check is particularly effective in verifying redundance management schemes or the system's proper reaction to failures. That is, given that a system failure is sensed, does the switch to a redundant element occur in an orderly manner?

The output from a program simulated in one of these languages normally allows, as a minimum, a trace of the simulator elements or blocks through which the data has traveled. This trace can verify the designer's intended data flow. Also, the time of arrival of the data at each model element can be continuously updated and evaluated for data delay, skew, and late and early arrivals. With these minimum outputs the systems designer can perform a great deal of "before the fact" debugging, where the potential savings in time and money are greatest. The data required to simulate a system at this level is normally neither great in volume nor detailed. Certainly the model need not be more detailed than the "shift register" or "arithmetic unit" level, and even that level of detail may not be required to obtain meaningful results.

Some of the functional requirements expected to be obtained by varying workloads for a single model of a multiplex system are as follows:

- Average system bit rate requi ements
- Maximum system bit rate requirements
- Average system data transfer capacity
- Maximum system data transfer capacity
- Average system data transfer reserve capacity
- Minimum system data transfer reserve capacity
- Maximum data transfer delay
- Average data transfer delay
- System reaction to faults

If several different multiplex systems are modeled by changing the variables of simulation, and if these models are exercised with varying workloads, then a more comprehensive "simulation experiment" will have been performed. In such a case, the following types of outputs may be obtained:

- Optimal number of remote terminals
- Optimal placement of rem te terminals
- Best system bit rate requirements

(2) Operational Outputs

Once a final system configuration has been tentatively selected and its correct logical behavior established, more detailed outputs may be obtained from the simulation model. The resultant information collected concerns how well the system operates rather than simply "does it work?" This data can provide specific information which can be used for "tuning" the design for optimal performance. Examples of outputs containing this more detailed information are the following:

- System performance with "worst-case" workloads
- Average and maximum utilization of various system resources:

  --bus utilization

  --control unit utilization

  --terminal unit utilization
- Probability of lost data
- Probability of undetected erroneous data

- Probability of late data
- Average and maximum data skew and jittering
- Nominal and worst-case sync behavior
- Detailed system queuing, priority, and interrupt behavior

(3) Other Outputs

The multiplex system designer must, by one means or another, provide additional information regarding a candidate multiplex system design. Although such information must be an output of his design study, it is not necessarily a direct output of MUXSIM.

Examples of these "other" outputs follow:

- System cost
- System weight
- System size
- System reliability
- System maintainability
- System performance/cost
- System power requirements
- Internal subsystem layout preferences

Because these outputs require information which may be difficult to quantify or otherwise incorporate into MUXSIM, they are not presently contemplated as direct outputs from MUXSIM. However, it is expected that the existing MUXSIM outputs can be used, together with other information, to produce these outputs after additional manual or automatic manipulation. As MUXSIM evolves, it may be possible to include some of these outputs as direct outputs of its future versions.

c. MUXSIM Implementation and Design Recommendations

The main purpose of the MUXSIM definition phase was to produce a functional definition or description of MUXSIM. Ideally such a definition should specify what functional capabilities MUXSIM should have, but not how they should be implemented. However, it has been noted previously that the MUXSIM functional definition has been significantly influenced by a knowledge of what the intended MUXSIM host system was to be (i.e. the DEC PDP-11/45). This knowledge played a role in restricting the scope of MUXSIM.

Other studies conducted during the MUXSIM definition phase, discussed earlier, led to a set of recommendations for MUXSIM implementation and design procedures. Although these are not truly functional requirements, it was felt that they were valid and significant inputs for the MUXSIM design phase. They are stated below.

- To reduce MUXSIM initial development costs and to facilitate later changes, the highest level implementation language feasible should be used. Specifically, GASP and/or Fortran should be used as primary implementation languages for the Simulator.

- The test bed mode of MUXSIM operation should be considered as separate from, and independent of, other modes of operation.

- The non-test-bed portion of MUXSIM should be considered mainly as software, with little or no special hardware required.

- To ensure MUXSIM feasibility and relevance, MUXSIM specifications should be developed in parallel with a small-scale simulation exercise to model and simulate those areas of the B-1 EMUX system which have, in hindsight, caused the most design problems, and wherein early simulation could apparently have done the most to alleviate those problems. An example of such a problem area is that of redundancy management. MUXSIM specifications should be developed in part by generalization of the small-scale simulation model developed for B-1 EMUX analysis.

SECTION IV

PHASE 2 - MUXSIM DESIGN

1. INTRODUCTION

The MUXSIM design phase was basically a three-step process which, starting from a MUXSIM functional description and requirements, proceeded to the development of a MUXSIM design specification. This process is illustrated in flow chart form in Figure A-13. The functional description and requirements which served as the input to this phase were produced as the principal output of the preceding definition phase.

This section consists mainly of a discussion of the How, What and Why of MUXSIM design; how the process was conducted, what design resulted, and why various design decisions were made. However, for the sake of brevity this discussion will be in summary form only. The reader interested in full detail may find it in the Second Interim Technical Report (ref. 6), which consists of four volumes containing about 750 pages altogether.

In general the design process was straightforward, and resulted in a design consistent with both the input functional requirements and the associated design and implementation recommendations. The only exceptions to this were two: (1) the requirement for a real-time or test-bed MUXSIM mode was dropped, with Air Force approval; and (2) certain MUXSIM outputs were added which are useful in the multiplex system production process (i.e., in software or hardware development) but which are not simulation outputs in a strict sense. The additional outputs were easily provided as a by-product of existing MUXSIM models and processes, and serve to increase its value considerably in real system developments.

The partitioning of design tasks was such that Tasks 1, 2 and 3 in Figure A-13 were necessarily executed sequentially, but much of the detail design of subsystem (done in Task 2) was accomplished in parallel. This particular organization of the design effort was quite effective in reducing overall design time. In the following sections the how, what and why of the three identified design tasks are discussed separately.

2. TASK 1 - SYSTEM FUNCTIONAL DESIGN

The MUXSIM functional definition produced in Phase 1 described what capabilities MUXSIM should have and what it was to be used for, but it did not adequately describe how MUXSIM was to be used. In short, it did not provide a well-defined MUXSIM-User interface definition; and this was a significant omission, because ease of use was an evident MUXSIM design goal. Also there were certain other weakly defined areas, notable among them being the area of test mode requirements.

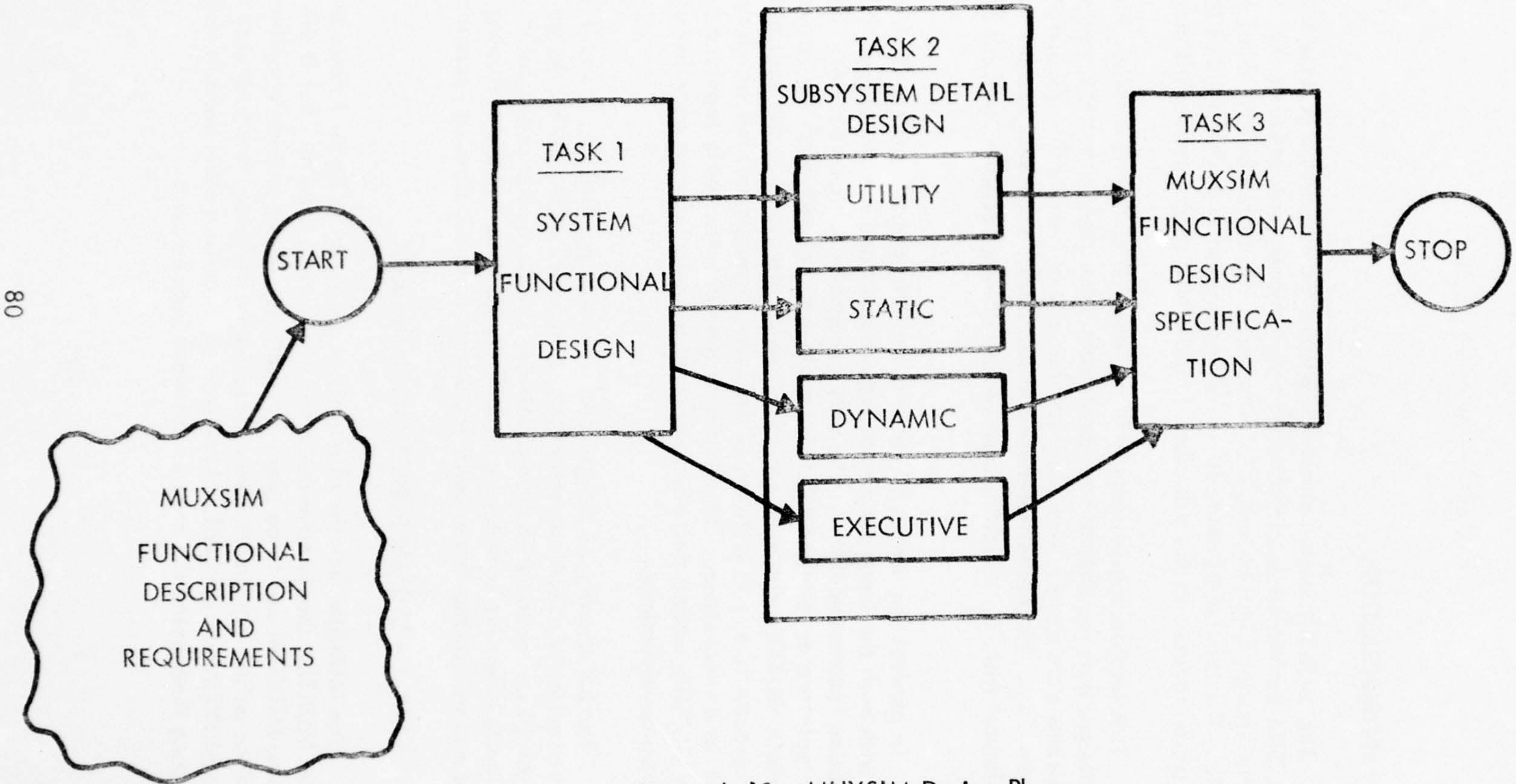


Figure A-13. MUXSIM Design Phase

The latter problem was quickly resolved by the deletion of MUXSIM test mode requirements, with Air Force agreement. The former problem was resolved by development of a MUXSIM use model together with a profile of a typical MUXSIM user, and these led to a set of requirements for the MUXSIM-user interface.

The MUXSIM use model developed was that for a typical simulation experiment, and a simplified version of it is given in Figure A-14. It contains five steps, which are discussed below:

Step 1 – System Definition

In this step the system to be studied is defined. A typical definition will take the form of an annotated block diagram. This block diagram need not be precise, serving mainly as a loose semiformal description of what is desired.

Step 2 – Model Building

In this step the following must be specified in a precise manner, suitable for input to a computer:

- System configuration (specification of system components)
- System sequencing and control
- System workload
- System status and environment (failures of system components, errors in system inputs)
- System monitoring and outputs (system performance data collection, data analysis, and output formats).

Step 3 – Simulation

At simulation time, items such as the following must be specified in a precise manner:

- Simulation run mode (e.g., with or without trace)
- Simulation run termination conditions
  - End after a certain amount of simulated time
  - End after a certain amount of elapsed time

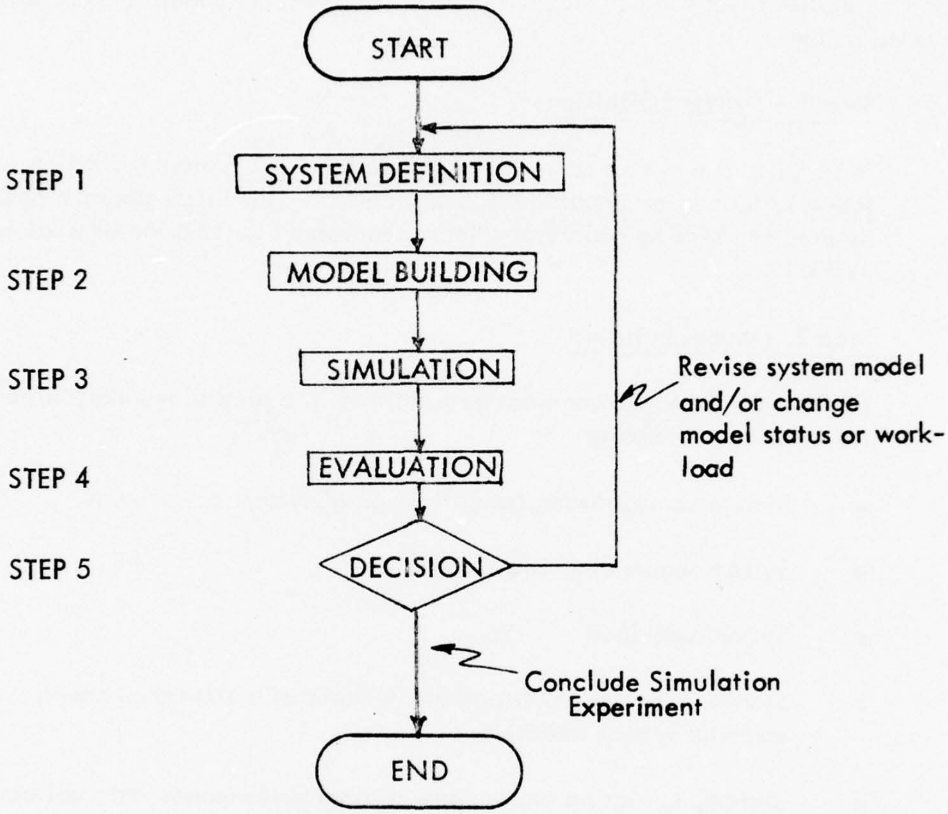


Figure A-14. MUXSIM Use Model for a Simulation Experiment

- End after a certain logical condition achieved
- End after certain error conditions
- End after a certain amount of output generated

- Other simulation run options
  - Trace mode (selective, full)
  - User intervention modes (user switch settings, etc.)
  - Selected output devices (e.g., printer or display terminal)

Various other options can be imagined, although they are not necessarily feasible. For example, the user might have a choice of compilers which optimize on either compiler time or run time, as is the case with the IBM 360 Fortran G and H compilers and with IBM 360 PL/1 compilers.

Step 4 - Evaluation

Following a simulation, evaluation of the results of the run is normally accomplished. Some of the evaluation can be done automatically, involving programmed analysis and display of raw data obtained as output from the simulation. The more of the evaluation that can be done automatically, the easier the simulator is to use, but an extensive automatic evaluation capability can add substantially to the simulator cost.

Step 5 - Decision

The exact nature of the decision step depends on the use mode of the simulator. If the simulator is being used primarily for research studies of system performance, and sensitivity to variation of certain system parameters, then the decision after a run has to do with what parameters to vary next and in what manner. However, if the question revolves around which of two possible systems is best for a specific development program, then the exercise is concluded after both systems have been simulated for the same workload and error/failure environments and the better of the two selected. Although the decision step can be partially automated, human judgment will probably be the principal ingredient here for the foreseeable future of MUXSIM.

In addition to the use model, a typical MUXSIM user was defined in terms of computer and multiplex system design background requirements. Since these assumed requirements are covered specifically in Appendix II, they will not be repeated here. Taken together, the MUXSIM use model and the typical user profile provided a basis for design of a MUXSIM which was "easy to use".

A number of other design requirements were explicitly identified as part of Task 1 efforts. These included:

Design Requirements

- Models should be valid and realistic, and both aspects should be demonstrable.
- Workloads should be demonstrably realistic, or else real workloads.
- Outputs should be useful, meaningful, and in easily interpreted formats.
- Cost of simulation runs should not be excessive.
- MUXSIM user training requirements should be minimal.
- MUXSIM implementation should be flexible, to allow easy change or modification.
- MUXSIM design should be compatible with the capabilities of its planned host computer system (the DEC PDP-11/45).
- MUXSIM design should be compatible with a reasonably low cost implementation effort.
- MUXSIM should be, in the fullest sense, cost-effective.

Although many of these requirements fall into the category of common sense, many simulators have been developed which do not meet them. For example, some simulators have been built which are so hard to use that they are in fact not used at all, while others are so expensive to run that economics prohibit their use. Recently, the highly publicized "limits to growth" simulations have been strongly criticized on the grounds that the models implemented are invalid or that input data is unrealistic. It was a goal of MUXSIM design to avoid all the above simulation pitfalls if possible.

Guided by the MUXSIM functional definition and with the user interface and the above design requirements established, the Task 1 effort concluded with the specification of the MUXSIM system functional design. This describes MUXSIM as a system consisting of four major subsystems, as shown in Figure A-15. The four subsystems and their functions are:

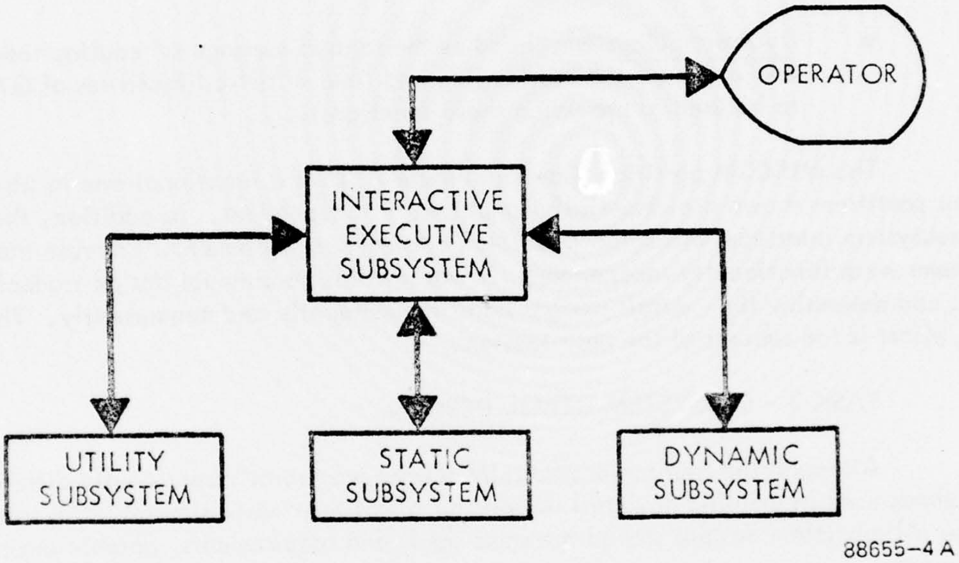


Figure A-15. MUXSIM Modular Software Structure

- Executive Subsystem: permits operation from a CRT or TTY terminal. Has an interactive section which can be switched on/off, to aid the user and facilitate learning the simulator operations. Controls other subsystems.

- Utility Subsystem: manages the signal flow list and extracts the simulator inputs (workload) from it. Uses the signal flow list to check the Equipment Complement for completeness, flagging any equipment and signal deficiencies.

- Static Subsystem: handles the remote terminal loading task, as well as the fixed-telemetry format message structure and bus loading and utilization computations.

- Dynamic Subsystem: handles the random message scheduling task, and computes the bus loading and time statistics (facilities of GASP to be used in providing these functions).

The MUXSIM partition shown in Figure A-15 is a functional one in which the four partitions shown are essentially functionally independent. In addition, the inter-subsystem interfaces are relatively clean and easy to implement. Because the four subsystems were functionally independent, it was possible to conduct design tradeoff studies and determine their detail designs both independently and concurrently. This design effort is the subject of the next section.

3. TASK 2 - SUBSYSTEM DETAIL DESIGN

Although the four major MUXSIM subsystems are all functionally different, the methods used in arriving at detail designs for all of them were similar. This was because all subsystem designs shared the same goals and requirements, notable among which were:

- Need to fit on a small computer system (the DEC PDP-11/45)

- Need to produce meaningful and useful outputs

- Need to use real or realistic inputs

- Need to include real or realistic models

As a result, the design efforts were all characterized by a series of small probe coding exercises, each of which was intended to produce meaningful and useful results related to some aspect of an existing real-world multiplex system design problem. For most of the exercises, the real problem was also the same, namely the design of the U. S. Air Force Avionics Laboratory DAIS Multiplex Core Element. Thus, for example, the Utility subsystem exercises focused on the actual signal flow list for the A-7D aircraft and suitable massaging thereof, while the Static Subsystem exercises focused on data transfer models and techniques compatible with the Air Force multiplex standard MIL-STD-1553. The thrust was to develop specific programs which clearly produced useful and realistic results and would fit on a small system, and to generalize thereon later. This method was suitable for developing and testing software techniques and algorithms, and for determining specifics of inputs and outputs; moreover it provided an environment which stimulated ideas for new MUXSIM functions, and ways to implement them. A major benefit was that the method provided a reliable basis for subsequent MUXSIM sizing exercises (required number of lines of source code, number of object code, size of data base, etc.) In brief, the design method used could be described as a piecemeal software breadboard technique. Some specifics of the individual subsystem "breadboards" are covered next.

a. Utility Subsystem

The Utility Subsystem breadboard consisted of a collection of Fortran programs and subprograms (about 1300 lines of source code). They massaged the Signal Flow List data in various ways to facilitate LRU-to-terminal assignments, and to verify the validity of the data and of the particular LRU-to-terminal assignments that were made. The Signal Flow List used was the actual one for the A-7D aircraft; and it was initially available on punched cards. It contained about 2650 signals, and each signal had a number of associated parameters. Altogether the A-7D signal flow list comprised a card data base of about 10,700 cards.

b. Static Subsystem

The Static Subsystem breadboard consisted mainly of a synthetic signal flow list generator, plus a program which mechanized a particular bus command and control model that was initially driven by the synthetic signal flow list. Subsequently, the program was modified so that it could be driven either by the synthetic signal flow list, or by the real signal flow list as filtered by the Utility Subsystem. A number of versions of this program were produced, in which various bus control models were tested, output formats developed, etc. The program, called DASIM, was written in Fortran, and it comprised about 1300 lines of source code. It consisted mainly of a three-stage sort/map process, as illustrated in Figure A-16.

c. Dynamic Subsystem

The "breadboarding" done in connection with the Dynamic Subsystem differed from the other efforts in that it was mainly an evaluation of GASP, aimed at ensuring its suitability as a vehicle for implementation of MUXSIM dynamic models. As has been noted elsewhere, MUXSIM dynamic models are discrete-event type simulation models which are useful in determining optimal bus command and control disciplines, optimal multiplex system redundancy management schemes, etc.

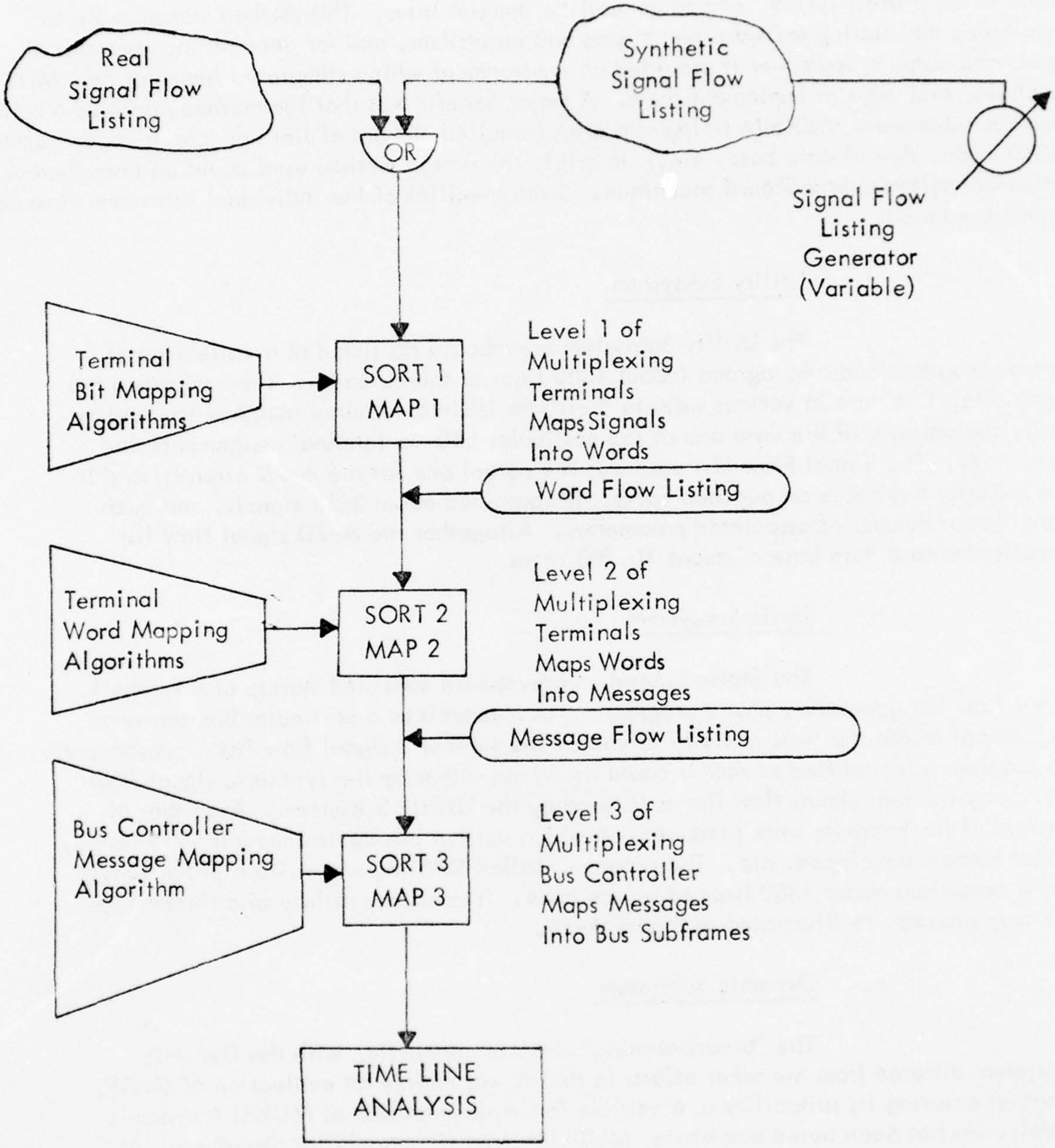

Real Signal Flow Listing
Synthetic Signal Flow Listing
OR
Signal Flow Listing Generator (Variable)
Terminal Bit Mapping Algorithms
SORT 1 MAP 1
Level 1 of Multiplexing Terminals Maps Signals Into Words
Word Flow Listing
Terminal Word Mapping Algorithms
SORT 2 MAP 2
Level 2 of Multiplexing Terminals Maps Words Into Messages
Message Flow Listing
Bus Controller Message Mapping Algorithm
SORT 3 MAP 3
Level 3 of Multiplexing Bus Controller Maps Messages Into Bus Subframes
TIME LINE ANALYSIS

Figure A-16. DASIM Flow Chart

A GASP II card deck (about 1500 cards) was obtained from the GASP developers/supporters (Pritsker and Associates), and made operational on the Harris Datacraft. Although GASP II is Fortran-based, a number of changes were required to perform the above. Thereafter, GASP II was used to develop and run several models typical of the simpler MUXSIM dynamic models that were anticipated. These exercises served to verify that GASP II was an appropriate choice for implementation of MUXSIM Dynamic Subsystem, because: (1) it made possible rapid development of dynamic MUXSIM models; and (2) it was appropriate for use on a minicomputer system.

d. Executive Subsystem

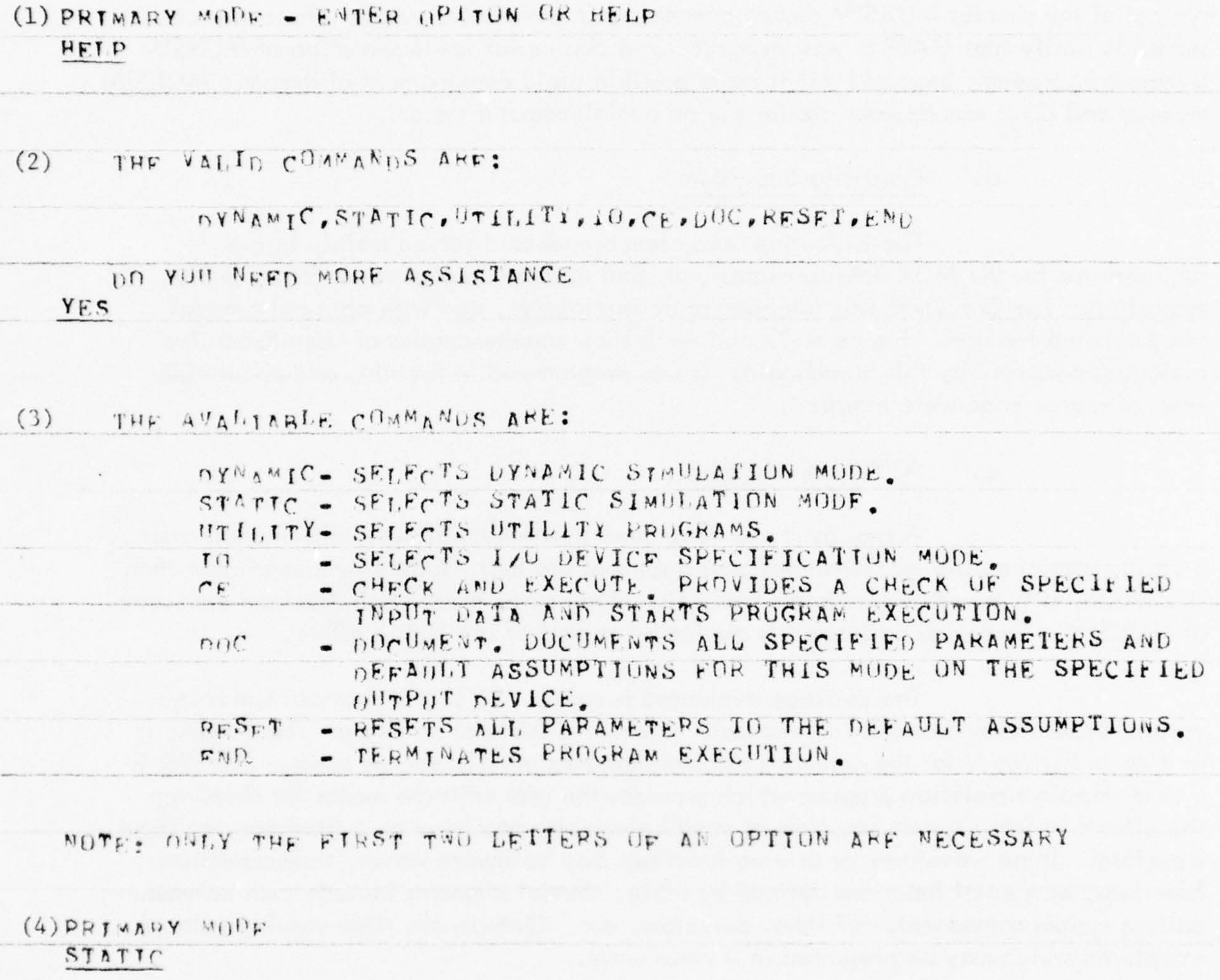
The Executive Subsystem breadboard served mainly to clarify requirements for the MUXSIM-user interface, and to develop algorithms for its implementation. The Executive was intended to be interactive, and with optional tutorial and coaching features. Figure A-17 and A-18 show some examples of User-Executive dialogues produced by this breadboard. It was programmed in Fortran, and about 1000 lines of source code were required.

e. ASYSTD Evaluation

Although it was concluded in the MUXSIM definition phase that certain types of multiplex system designer questions are best handled by means other than simulation, it is true that simulators now exist which cover areas excluded from the scope of MUXSIM. One such package was evaluated as part of the Task 2 effort.

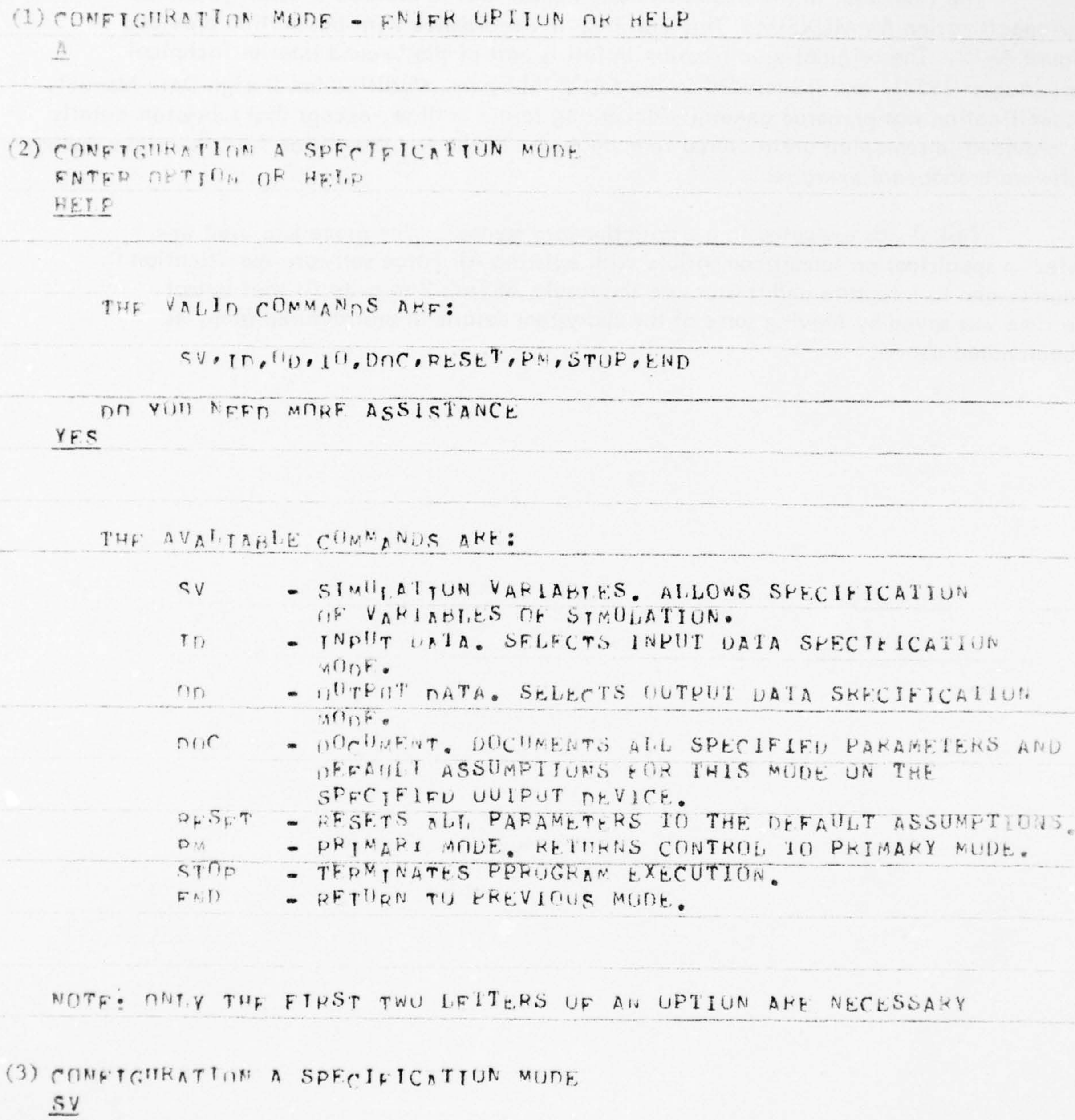
The package evaluated is called ASYSTD (Advanced Communications System Time Domain Techniques). It was available at no charge from NASA, is written in Fortran V for the Univac 1108, and consists of 9133 source cards. ASYSTD is a time domain simulation program which provides the user with the means for observing the effects which a system (cascade of model elements) may have on a time-varying input waveform. Input waveforms or driving functions may be square waves, transcendental functions, or special functions defined by table. Model elements include such communications system components as filters, detectors, etc. Outputs are time-varying voltage waveforms which may be presented in various ways.

It was concluded that there was no immediate application for ASYSTD techniques in the treatment of bus traffic handling problems, which are a primary focus of MUXSIM. The language of ASYSTD was found to be simple and efficient; ASYSTD use requires only a minimal amount of code from the user. Finally, it was found that ASYSTD computer runs are expensive, ranging from $10 for the simplest runs to an estimated several hundred dollars for the more complex models.



```
(1) PRIMARY MODE - ENTER OPITON OR HELP
    HELP

(2)     THE VALID COMMANDS ARE:

              DYNAMIC,STATIC,UTILITY,IO,CE,DOC,RESET,END

        DO YOU NEED MORE ASSISTANCE
    YES

(3)     THE AVAILABLE COMMANDS ARE:

              DYNAMIC- SELECTS DYNAMIC SIMULATION MODE.
              STATIC - SELECTS STATIC SIMULATION MODE.
              UTILITY- SELECTS UTILITY PROGRAMS.
              IO     - SELECTS I/O DEVICE SPECIFICATION MODE.
              CE     - CHECK AND EXECUTE. PROVIDES A CHECK OF SPECIFIED
                       INPUT DATA AND STARTS PROGRAM EXECUTION.
              DOC    - DOCUMENT. DOCUMENTS ALL SPECIFIED PARAMETERS AND
                       DEFAULT ASSUMPTIONS FOR THIS MODE ON THE SPECIFIED
                       OUTPUT DEVICE.
              RESET  - RESETS ALL PARAMETERS TO THE DEFAULT ASSUMPTIONS.
              END    - TERMINATES PROGRAM EXECUTION.

    NOTE: ONLY THE FIRST TWO LETTERS OF AN OPTION ARE NECESSARY

(4) PRIMARY MODE
    STATIC
```

Figure A-17. Primary Mode User Interaction

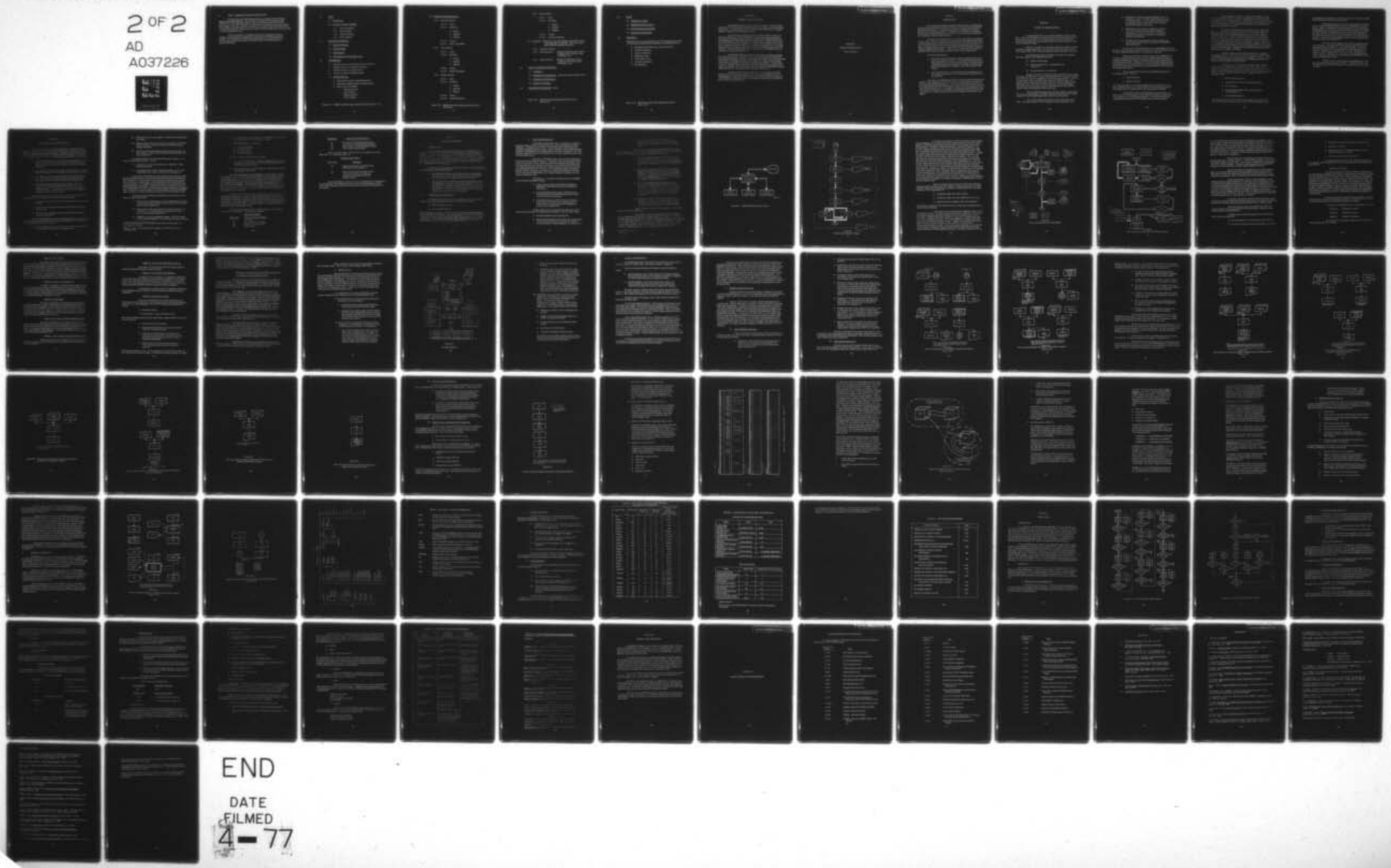


```
(1) CONFIGURATION MODE - ENTER OPTION OR HELP
    A

(2) CONFIGURATION A SPECIFICATION MODE
    ENTER OPTION OR HELP
    HELP

       THE VALID COMMANDS ARE:

            SV,ID,OD,IO,DOC,RESET,PM,STOP,END

       DO YOU NEED MORE ASSISTANCE
    YES

       THE AVAILABLE COMMANDS ARE:

            SV       - SIMULATION VARIABLES. ALLOWS SPECIFICATION
                       OF VARIABLES OF SIMULATION.
            ID       - INPUT DATA. SELECTS INPUT DATA SPECIFICATION
                       MODE.
            OD       - OUTPUT DATA. SELECTS OUTPUT DATA SPECIFICATION
                       MODE.
            DOC      - DOCUMENT. DOCUMENTS ALL SPECIFIED PARAMETERS AND
                       DEFAULT ASSUMPTIONS FOR THIS MODE ON THE
                       SPECIFIED OUTPUT DEVICE.
            RESET    - RESETS ALL PARAMETERS TO THE DEFAULT ASSUMPTIONS.
            PM       - PRIMARY MODE. RETURNS CONTROL TO PRIMARY MODE.
            STOP     - TERMINATES PPROGRAM EXECUTION.
            END      - RETURN TO PREVIOUS MODE.

    NOTE: ONLY THE FIRST TWO LETTERS OF AN OPTION ARE NECESSARY

(3) CONFIGURATION A SPECIFICATION MODE
    SV
```

Figure A-18. Static Simulation User Interaction

AD-A037 226 HARRIS CORP MELBOURNE FLA ELECTRONIC SYSTEMS DIV F/G 9/4
MULTIPLEX SIMULATOR DESIGN STUDY.(U)
JAN 77 J A GRACIA, P J KNOKE F33615-73-C-1172
UNCLASSIFIED AFAL-TR-76-201 NL

2 OF 2
AD
A037226



END
DATE
FILMED
4-77

4. TASK 3 - FUNCTIONAL DESIGN SPECIFICATIONS

The final task in the MUXSIM design phase was to produce a detail functional design specification for MUXSIM. This was done in accordance with the outline provided in Figure A-19. The original specification in full is part of the Second Interim Technical Report (August 1974), and is included in the MUXSIM System Modification Design Data Manual. The specification was prepared generally according to the outline, except that subsystem details were provided in somewhat unstructured form by means of the documentation from the probe coding or software breadboard exercises.

Task 3 was executed in a straightforward manner. The procedure used was to select a specification format compatible with existing Air Force software specification standards, and to integrate and restructure the results of Task 2 so as to fit that format. Some time was saved by leaving some of the subsystem details in unstructured form, as has been noted above.

1.0 SCOPE

1.1 Identification

1.2 Functional Summary of MUXSIM

1.2.1 Executive Subsystem

1.2.2 Static Subsystem

1.2.3 Dynamic Subsystem

1.2.4 Utility Subsystem

2.0 APPLICABLE DOCUMENTS

2.1 Radiation Documents

2.2 GASP Documents

2.3 DEC Documents

2.4 Other Documents (Military Specs, etc.)

3.0 REQUIREMENTS

- General description of host system hardware and software
- MUXSIM interfaces with peripheral equipment
- MUXSIM interfaces with other programs
- General description of MUXSIM functions

3.1 MUXSIM Definition

- Requirements imposed by interfaced equipment
- Requirements imposed by other interfaced programs
- Major functions of MUXSIM
  - Executive subsystem
  - Static subsystem
  - Dynamic subsystem
  - Utility subsystem

Figure A-19. MUXSIM Functional Design Specification Outline (Sheet 1 of 4)

3.2 Detailed Functional Requirements

3.2.1 Executive subsystem

3.2.1.1 Inputs

3.2.1.2 Processing

- Purpose
- Approach
- Diagrams

3.2.1.3 Outputs

3.2.1.4 Special requirements

3.2.2 Static subsystem

3.2.2.1 Inputs

3.2.2.2 Processing

- Purpose
- Approach
- Diagrams

3.2.2.3 Outputs

3.2.2.4 Special requirements

3.2.3 Dynamic subsystem

3.2.3.1 Inputs

3.2.3.2 Processing

- Purpose
- Approach
- Diagrams

3.2.3.3 Outputs

3.2.3.4 Special requirements

Figure A-19. MUXSIM Functional Design Specification Outline (Sheet 2 of 4)

3.2.4 Utility subsystem

3.2.4.1 Inputs

3.2.4.2 Processing

- Purpose
- Approach
- Diagrams

3.2.4.3 Outputs

3.2.4.4 Special requirements

3.3 Adaptation (Factors associated with a different choice of host system or operating system for MUXSIM. These factors affect scope of MUXSIM operational functions.)

3.3.1 General environment

3.3.2 System parameters (Constant describing maximum number of signals in SFL, maximum number of terminals, etc.)

3.3.3 System capacities (Number of simultaneous displays, host system core and mass storage requirements, etc.)

4.0 QUALITY ASSURANCE PROVISIONS

4.1 Introduction

4.2 Development Test Requirements (All levels except acceptance level)

4.3 Acceptance Test Requirements

4.4 Summary of Test Program

5.0 PREPARATION FOR DELIVERY (N/A)

Figure A-19. MUXSIM Functional Design Specification Outline (Sheet 3 of 4)

6.0 NOTES

6.1 MUXSIM size estimates

6.2 MUXSIM Specification Process

6.3 Modular Programming Guidelines

6.4 Documentation Requirements

7.0 APPENDICES

(Requirements which are contractually a part of the specification but which, for convenience of specification maintenance, are incorporated herein)

- Mathematical derivations (e.g., of test case results)
- Alternative algorithms
- Summary of equations
- Definitions of terms
- DAIS Technical Notes
- B1 technical documents
- SFL descriptions

Figure A-19. MUXSIM Functional Design Specification Outline (Sheet 4 of 4)

SECTION V

SUMMARY AND CONCLUSIONS

The MUXSIM development plan, covered in Section II, called for a three-phase effort, in which the phases were MUXSIM definition, design, and implementation, respectively. This appendix has covered the definition and design phases. A separate appendix is devoted to the implementation phase.

The major problem solved in the definition phase (covered in Section III) was that of scoping; i.e., what was MUXSIM to include, what to exclude, and why. This problem was simplified by the assumption (since proven invalid) that MUXSIM was to be implemented on a DEC PDP-11/45 minicomputer system. Another useful scoping device was a decision to focus MUXSIM, as a multiplex system designer's analysis tool, on those significant design questions for which answers were required early in the design process, and for which answers could not be conveniently obtained by means other than simulation. The major output of the definition phase was a set of MUXSIM functional requirements, covered in Section III, 5.

The MUXSIM design phase, covered in Section IV, was done in a straightforward three-step process. The first step was a functional partition of MUXSIM into four independent subsystems; the next, a detail functional design of each of the subsystems; and the last, an integration of the four detail designs into a single functional design specification per applicable military software specification standards. A probe coding or software "breadboarding" approach was used in the second step, and found to have a number of advantages. The major output of the design phase was the MUXSIM functional Design Specification, the outline for which is given in Section IV, 4.

A conclusion of the two-phase effort was that a cost-effective MUXSIM could be functionally defined, and that a version of it could be designed so as to be implementable on a DEC PDP-11/45 host system. The evidence for this conclusion lies in the MUXSIM functional definition and the MUXSIM functional design specification, both of which are described herein.

APPENDIX B

MUXSIM IMPLEMENTATION

PHASE III REPORT

SECTION I

INTRODUCTION

The third phase of Multiplex Simulator Design Study was an implementation of the basic parts of the Software Simulator which were conceived in phase one and designed in the second phase. The requirements outlined in the Statement of Work for Phase Three were for the construction and verification of a Simulator's Main Software Structure and a set of models which are in line with current-day hardware implementation concepts for Avionics Multiplex Systems.

The object of the implementation was to prove the feasibility and usefulness of such a tool, as the basis for development of concepts for more sophisticated practical multiplex systems and as the basis for development of future, more refined versions of the tool. An initial evaluation of that which was implemented considers the simulator as a software package which does computer-aided design and design verification of digital information transfer/multiplex system. It appears to be a valuable aid for an organized approach at specifying and designing multiplex systems for diverse applications.

Two obvious advantages are:

- Quick-look evaluations of different systems by allowing a man, interacting with a computer terminal, to call the different models and get rapid quantitative answers which are derived from a fairly sophisticated and detailed data base representation of his problem, thereby greatly reducing the probability of a judgment error in his evaluation of the problem.

- Low-cost documentation of the potential system selected down to the pin assignment level for use in system and subsystem specifications and documentation.

The scope of the implementation effort was to build the basic framework and implement three of the eight static and one of the two dynamic models which were defined during the MUXSIM design phase. The models were defined as representative of the current hardware implementation philosophy and are considered typical of those with which MUXSIM needs to cope. In retrospect, the framework implemented makes model construction easy; therefore, all ten models were implemented instead of the four which were planned. It turned out to be a minimal effort to implement the additional static models.

SECTION II

HISTORY OF IMPLEMENTATION

The Implementation Phase was initiated immediately after the Design Phase. Since the implementation was conducted by the team that had conducted the Conceptual and Design effort, it was handled more as an extension to the prior effort, thereby facilitating a larger interplay between the results of the prior efforts and the findings of this phase.

Some major concepts were streamlined, particularly the data base hand-off between the software subsystems. The basic time frame of this effort was July 1974 through October 1975. The final acceptance test was conducted in June 1976.

The Implementation effort was handled in three parts, as with many other major software system implementation efforts:

(1) Software System Design

(2) Software Program Design, Implementation and Verification

(3) System Integration and Verification

However, the effort was impacted by the unavailability of the intended host system, an AFAL PDP-11, some unanticipated difficulties in using the operating system RSX-11D, and the unavailability of host system support personnel. A solution was generated to allow use of AFAL's DEC System-10 as the host system. This change caused a program disruption which resulted in a temporary implementation phase effort reset. In an effort to minimize cost impact, parts of the Software System Design were left the same, since the DEC System-10 can execute PDP-11 software without major penalties in time of execution.

The actual partition between the design effort and the implementation effort was not cleanly divided. The design effort was carried forward and reflected somewhat in the first implementation task, the Software System design.

Also, a probe-coding effort had been conducted during the design phase. The main requirements which instigated this probe-coding effort were:

- Establishment of the physical size for MUXSIM, since the intended host system was a PDP-11/45 minicomputer which has limited capability. It was considered a risk and action was therefore required to assert that this effort would be feasible while using that host system.

- Establishment of some degree of confidence in the feasibility of the functional (user) and logical (software) requirements of the various subsystems. It was felt that some growth in experience was necessary to establish a working system definition.

- Establishment of the best possible functional, logical, and physical parameters description for inclusion in the MUXSIM System Specification.

The result of this probe-coding effort was an understanding of the translation of the user requirements to software requirements. It also helped jell and narrow the scope of MUXSIM implementation to a feasible and usable tool.

Another result which did not become apparent to the MUXSIM team until later in the implementation phase was that the probe-coding effort served as a software breadboard and therefore the emerging system was more polished and refined than was anticipated. Attesting to this was the fact that the physical description parameters (lines of code) projected as a result of the probe-coding effort were almost twice those actually required; what'was accomplished was a larger task than had originally been scoped.

The two major divisions of the task were planned and executed as two separate entities. They are:

- User Ease Software

- Operation Software

An interface requirement was established between these two efforts early in the game, modified once when the change to DEC System-10 came along, and then not disturbed until the integration effort when the concept was finalized.

The integration effort was carried forward in two basic steps. An effort to marry one of the programs from the Operation Software was pursued early in the program to verify the integration approach. The bulk of this effort was expended toward the conclusion of the program, prior to final System Test.

Since the Operation Software is run program-by-program in pseudo-batch, the verification task of the individual program went a long way in aiding the verification of the system performance. Pseudo-batch in this sense means that, even after the system is integrated, the programs still function basically as a separate entity. This is attributed to the requirement for modularity which resulted in the programs being functionally isolated.

The functional requirements of this particular System, and for that matter even the Subsystems themselves, were such that the logical construction of modules lends itself to a progressive operation on the data base, where the end result of each modular step has significance to the user. For example, the Signal List (Data base) is mapped in remotely located terminals (remote terminal assignment). The output of this program is useful because it shows the designer the signal handling requirements for the remote terminal. The signals per terminal are then assigned to data words, and words are assigned to messages. Each one of these steps bears useful results because it gives the designer further system specification for his terminal design. These governing requirements presented the logical break points for the modular construction.

This effort, however, also set forth a requirement for progressive inter-program hand-off, which was best handled by a lower-level effort, longer time-frame program than was perhaps originally anticipated. In this way, the programs which were developed avoided expenditures of debugging awkward inter-program interfaces during the integration effort.

Five field trips were required during the DEC System-10 program. (Two earlier trips were conducted on the PDP-11/45 effort). The first DEC System-10 trip was primarily a facility familiarization trip and served as an introduction between Harris personnel and AFAL and AFAL support contractor personnel. However, since a considerable effort had already transpired, the following software was verified:

1. GASP IV Subroutine Library
2. User Interactive and Executive Subsystem Concepts
3. Utility Subsystem
4. Static Subsystem Programs which had been coded at the time of that trip.
5. Use of DEC Editing System

Also verified were the use of the portable remote telephone access terminal (TI Silent 700) and tape compatibility between Harris' facility and AFAL's system.

All programs were left resident in source file in the system, and could be edited and/or executed by means of the remote terminal.

During the remainder of the software program design and verification, the remaining MUXSIM programs were built and debugged on the Harris facility. After a set of programs had been completed, they were modified for the DEC System-10. This set was then shipped to AFAL, where it was installed on MUXSIM's assigned Disk Storage area in a source file by AFAL personnel. Harris personnel then proceeded to access these programs from Melbourne via the Remote Terminal and to direct successful compilation and execution of each program. This approach saved considerable travel and time in program debug and installation on the DEC System-10.

The initial system integration effort was also conducted remotely via this interface. The second, third, and fourth DEC System-10 trips were conducted at the finalization of the system integration effort. During this period the DEC System-10 was being upgraded to a dual CPU System. This resulted in a slightly longer integration effort, due to the DEC System-10 transitory nature. All programs were reverified and then integrated into the system. These trips also served as an initial debug of the User Manual and the actual system operational concept.

The fifth trip was for the purpose of final Acceptance test. This was the system demonstration and sell-off effort. The MUXSIM System Test Plan (Reference 10) contains the Acceptance Test Procedure which was used for this effort.

SECTION III

REQUIREMENTS FOR IMPLEMENTATION

The contractual requirements evoked on Harris Electronic Systems Division during the implementation phase were to provide the personnel, materials and facilities to implement the multiplex simulator system design. The overall program entailed the preparation of individual program modules, the integration of these modules into a simulator system and demonstration of the system. Specifically the requirements are as listed below:

- To develop and implement the detailed program modules from the system design accomplished under the prior phases. The modules were to be coded, debugged, and tested for use on the DEC System-10 (originally PDP 11/45).
- To integrate the individual modules into a simulator system. The system was to be tested to prove that the modules function together as a system.
- To develop a test plan for each of the modules and for the overall system. The test plan was to detail the exact steps to be performed in order to prove that the modules and system are operating correctly.
- To demonstrate that the simulator works as specified. This demonstration was to be on the DEC System-10 computer located in the Air Force Avionics Laboratory. The contractor was to integrate the simulator on the DEC System-10. The demonstration was to include the analysis of a proposed or working multiplex system.

Additional requirements are extracted from the efforts of the prior Define and Design phases. They are categorized into the following three divisions:

- Functional - those requirements which are specified from the user's point of view.
- Logical - those requirements which are specified from the programmer's point of view.
- Physical - those requirements which are specified from the computer operator's point of view.

With the above definitions in mind, the prime functional or user requirement of the simulator implementation phase was to construct a software simulation system, as prior defined and designed, which would be:

(1) easily modified and expanded to answer system-level questions generated by the dynamic environment of the multiplex world.

(2) effectively used in the early stages of a hardware design and development project.

(3) detailed enough to provide continuing value throughout the hardware project evolvement, particularly during the evaluation of system redesign requirements.

(4) easily learned and comprehended by people who need the tool, and doesn't require a highly trained software simulation specialist to use the tool effectively.

Two different additional functional requirements became apparent at the outset of the implementation effort. They are:

(1) requirement for a more text-oriented output, formatted in a more readable presentation.

(2) a requirement which called for a family of outputs, such as a text-oriented signal flow summary, formatting of bus messages, etc.

At the outset, the MUXSIM team had anticipated using only the bus traffic requirements data required by the signal flow summaries and which was hand-extracted from the signal list. It was, therefore, deemed necessary to expand the input requirements at this time for all signal data, including dictionaries which would provide the user with added text. This expanded input was necessary for the creation of user-readable formats for the various outputs. Another factor which was weighed at the time of initial implementation was the fact that planning and implementing some of this approach early, would result in a minimum impact to the overall program and enhance MUXSIM's usage. Therefore, this effort was factored in during this system implementation effort.

The following logical (programming) requirements were working constraints during the implementation phase:

(1) Software system transportability or machine-independence requirement. It had been specified that MUXSIM should enjoy a relatively high degree of machine-independence.

(2) Highly modular system which lends itself to easier construction and verification. It was desirable that the modules be functional in nature to enhance program modification.

(3) FORTRAN IV to be the programming language. GASP IV (General Activity Simulation Program)[1,2] was to be used for dynamic simulation.

[1] GASP IV Subroutine library can be purchased from Pritsker and Associates, Inc., West Lafayette, Indiana.

[2] Pritsker, A.A.B., The GASP Simulation Language, John Wiley & Sons, Inc., New York, 1974.

(4) The system to operate interactively for ease of operation, with optional coaching features to facilitate learning the system.

(5) The following subsystem partitioning

- EXECUTIVE Subsystem
- UTILITY Subsystem
- STATIC Subsystem
- DYNAMIC Subsystem

The following physical requirements were evoked:

(1) For economy in development, the modules (program) were to be coded, debugged, and tested on Harris' DPL (Datacraft 6024/5) machine prior to installation on AFAL's system.

(2) AFAL's Host System to be a DEC System-10. Originally the intended host system was a PDP 11/45.

With these requirements the team proceeded into the first effort, the design of the Software System. The resulting goal of the simulation software design effort was a set of modular programs which could be constructed and verified independent of other modules. One restriction evolved as a result of this partition. That is, most of the programs defined were sequential in nature; the output of the prior program becomes the input for the preceding program. The programming effort was best handled sequentially rather than in a parallel effort to avoid integration problems caused by awkward interfaces resulting from early definition. It proved to be best solved by the program actual low manpower expanded time effort.

The design phase defined eight static models and two dynamic models. The static models were defined to handle all the signal grouping and assignments which are representative of MUX hardware implementation grouping of signals. The static subsystem in general handles those computations associated with non-variant periodic transfer of information. The dynamic models handle those computations associated with stochastic time variant transfer of information. The dynamic models also can take into account the effects of the non-variant periodic transfer.

Of the eight static models defined, the effort originally was scoped to do three models: SA, SB and SE. The eight static models are:

STATIC MUX MODELS

Model Name	Information Transfer Discipline
SA	T/T Transfer
SB	T/C/T transfer (bit shuffling)
SC	Digital T/T, Discrete T/C/T
SD	Hybrid Transfer

Model Name	Information Transfer Discipline
SE	T/T Transfer with BCIU Broadcast Reception
SF	T/C/T Transfer with BCIU Broadcast Reception
SG	Hybrid Transfer with BCIU Broadcast Reception
SH	T/C/T Transfer (word shuffling)

Of the two dynamic models defined, only one was scoped for this effort (Model DA). The two dynamic models are:

DYNAMIC MUX MODELS

Model Name	Description
DA	Model of MUX system using demand-access information transfer disciplines.
DB	Model of MUX command-response information transfer disciplines, incorporating consideration of redundancy and fault-handling schemes.

After the MUXSIM main software structure was implemented it was determined that the additional Static Model construction was easier than anticipated,so all eight models were implemented. Both dynamic models were also implemented as agreed to on the Test Plan.

SECTION IV

MUXSIM CONSTRUCTION

1. INTRODUCTION

In order to adequately explain the construction of the MUXSIM system software structure, it must be viewed from the functional (user), logical (programmer), and physical (computer operator) points of view. Functionally, the points of view are: "What is it for and how is it used?" Logically, it is structured into two parts: that software which derives the desired results and that software which is for user ease or convenience. Physically, it is viewed from: How it runs, storage size, time for program execution, input requirements and output requirements.

The following two notes should be made of the specific simulator construction developed during this implementation phase:

- The interrelationship between the static and dynamic was not better explored because of lack of realistic data base, which contains the parameters defining the signalling requirements stochastically.
- The present hardware implementation concepts projected the relationship implemented. However, as new hardware concepts develop and consequently new models are generated, this relationship could change. The programs are modular and as long as interface requirements are respected they can easily be directly replaced with new ones or new programs can be inserted between existing ones.

2. FUNCTIONAL CONSTRUCTIONS

The functional or user view of the system is best described by surveying the system from two distinct vantage points posed by the following two questions:

- What is MUXSIM used for?
- How is MUXSIM applied?

The first question is answered by a brief description of the system's manipulation of the different pertinent users' parameters. The second question is treated by a description of a typical user's scenario. The prime object of both of these discussions is to present a useful overview. For interested personnel, the details of operation are covered in the MUXSIM User's Manual and MUXSIM System Modification Design Data Manual.

a. What is MUXSIM Used For?

This MUXSIM implementation effort was developed to address the design and design verification of digital data systems used for handling intra-aircraft information. It addresses any of a family of aircraft data transfer requirements where information originating from many sources, spatially separated, must be transferred and/or processed, then retransferred to an equally complex destination arrangement. The implementation is general in nature so that MUXSIM can be used for diverse applications where the above problem exists.

Essentially, the MUXSIM programs serve to link the gap between detailed analysis and prototype hardware. It provides a means of interplaying the pieces of the detailed analysis (such as update rate requirements, sampling requirements, data buffering requirements, bus data rate requirements, processing delay requirements) from a myriad of point-to-point signalling into a coherent requirement which can be verified for compatible performance by a computer prior to attempting a hardware development program, thereby preventing costly system errors from being implemented in hardware. For this reason the cost effectiveness of the simulator depends on its earliest use in the design or modification cycle.

The following is postulated as an example of the use of MUXSIM during a hardware development program.

- In the conceptual stages, past data bases developed by MUXSIM for similar systems are useful for bounding the requirements.
- As the development program matures, MUXSIM uses the emerging information to help develop the peculiar system's actual data handling requirements.
- As candidate systems are selected, they are evaluated to indicate specific relative meritsand problems of each. A final selection is based on how these advantages and problems affect the application.

The basic operation tasks to accomplish the above were defined during the prior conceptual and design phases of MUXSIM and were divided as follows:

- Data base management and accounting task
- Signal information grouping and handling task (representative of hardware implementation requirements such as remote terminal assignment, message assignment, bus schedule, etc.)

- Dynamic simulation of data transfer and processing requirements task to derive time delay statistics, data buffering requirements and other potential data queueing problems.

From those three operation tasks the MUXSIM software structure of four (three operation plus one user ease) subsystems emerged. They were also defined during the design phase,and their areas of responsibility are as follows:

- The Utility Subsystem manages the Signal Flow List and extracts the simulator inputs from it. It uses the information from the Signal Flow List to check the Equipment Complement for completeness, flagging any equipment and signal deficiencies.

- The Static Subsystem handles all the signal information grouping and handling such as Remote Terminal (RT MAP) loading task, the data bus word structuring (WORD MAP), the data bus message structuring (MESSAGE MAP), as well as the fixed-telemetry format message structure and average bus loading and utilization computations.

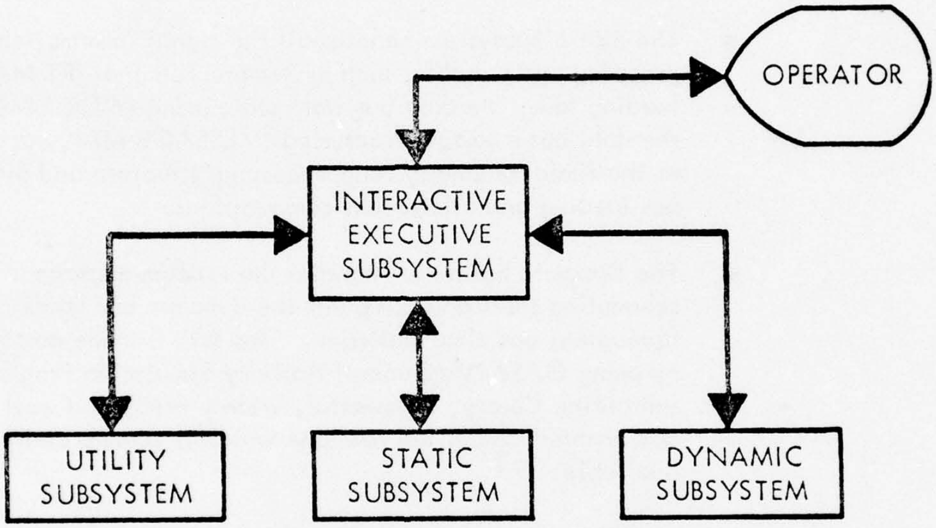
- The Dynamic Subsystem handles the random message scheduling tasks and computes the dynamic bus loading (queueing) and time statistics. This task is made possible by using GASP IV (General Activity Simulation Program) subroutine library, a powerful, widely used, and well documented simulation package which is commercially available.[1,2]

- The Executive Subsystem (User Ease Software) permits operation from a CRT or TTY terminal (console) and has an interactive section, with optional coaching, to aid the user and facilitate learning the simulator operations.

Figure B-1 depicts the above MUXSIM subsystem modular software structure and its interrelationships.

Functionally the three operation subsystems are sequenced as shown in Figure B-2 when a problem is being worked from start to end. It is, however, anticipated that other iterations will be useful in different circumstances. (For instance, once the Signal Flow Summary is derived from a fixed Signal List-Data Base ,it is speculated that the Utility Subsystem would be bypassed when simulating other Remote Terminal Configurations or running different Static Models.) The functional requirements are, however, more adequately described using the sequence shown in Figure B-2.

[1,2] op. cit.



88655-4 A

Figure B-1. MUXSIM Modular Software Structure

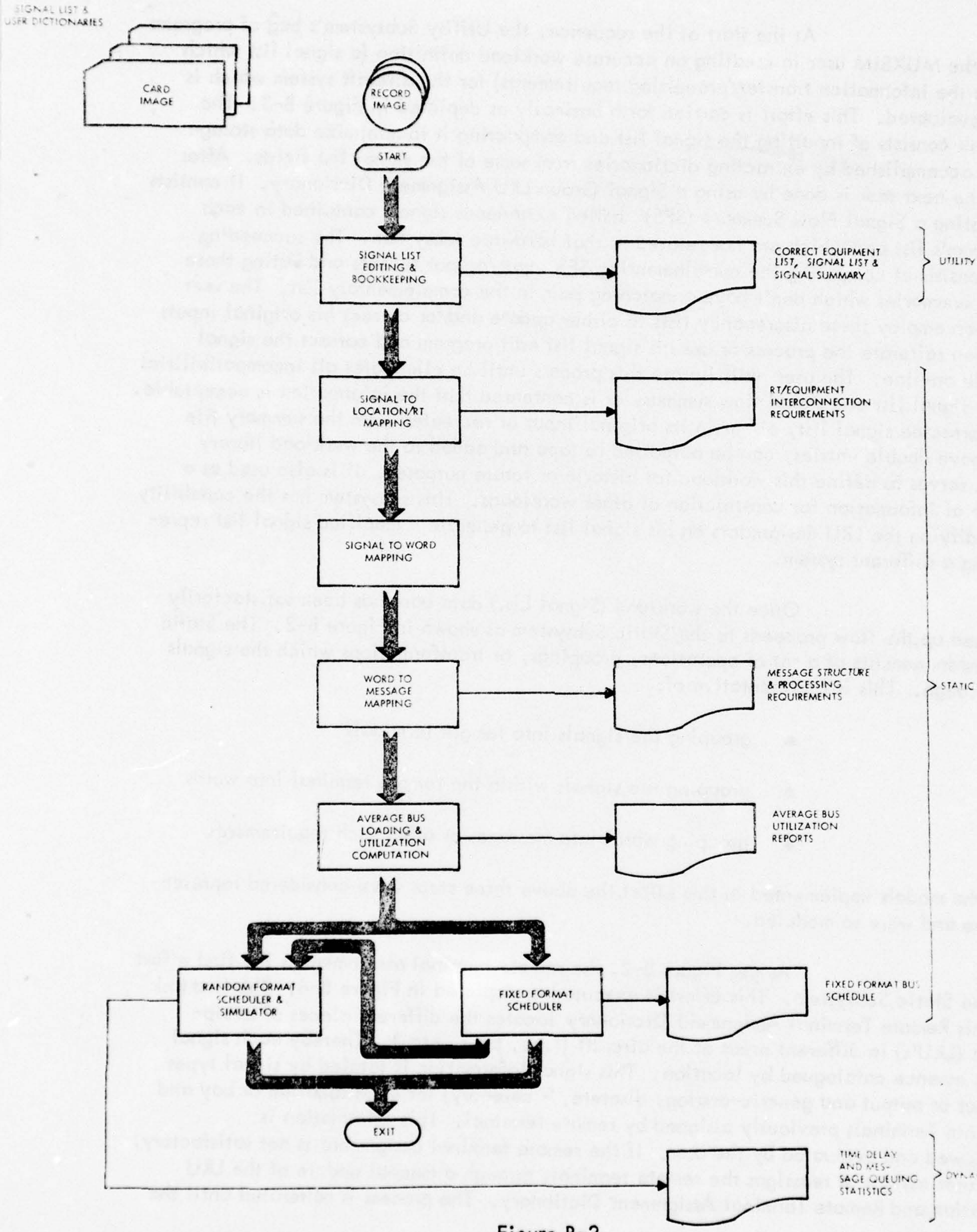

SIGNAL LIST & USER DICTIONARIES
CARD IMAGE
RECORD IMAGE
START
SIGNAL LIST EDITING & BOOKKEEPING
CORRECT EQUIPMENT LIST, SIGNAL LIST & SIGNAL SUMMARY
UTILITY
SIGNAL TO LOCATION/RT MAPPING
RT/EQUIPMENT INTERCONNECTION REQUIREMENTS
SIGNAL TO WORD MAPPING
WORD TO MESSAGE MAPPING
MESSAGE STRUCTURE & PROCESSING REQUIREMENTS
STATIC
AVERAGE BUS LOADING & UTILIZATION COMPUTATION
AVERAGE BUS UTILIZATION REPORTS
RANDOM FORMAT SCHEDULER & SIMULATOR
FIXED FORMAT SCHEDULER
FIXED FORMAT BUS SCHEDULE
EXIT
TIME DELAY AND MES-SAGE QUEUING STATISTICS
DYNAMIC

Figure B-2.
MUXSIM System Data Flow Diagram

At the start of the sequence, the Utility Subsystem's bag of programs assists the MUXSIM user in creating an accurate workload definition (a signal list which defines the information transfer/processing requirements) for the aircraft system which is being evaluated. This effort is carried forth basically as depicted in Figure B-3. The first task consists of inputting the signal list and compacting it to minimize data storage. This is accomplished by extracting dictionaries from some of the signal list fields. After that, the next task is done by using a Signal Group LRU Assignment Dictionary. It consists of creating a Signal Flow Summary (SFS) listing extraneous signals contained in each subsystem's list and which are not related to that hardware subsystem. The succeeding task consists of comparing the complementing SFS input/output records and listing those signal summaries which don't have a matching pair in the complementary list. The user can then employ these discrepancy lists to either update and/or correct his original inputs and then reiterate the process or use his signal list edit program and correct the signal list file on-line. The user will iterate this process until he eliminates all incompatibilities in the signal list and signal flow summary or is contented that the information is acceptable. The corrected signal list, either in its original input or recreated from the summary file to remove double entries, can be outputted to tape and added to the workload library which serves to define this workload for historic or future purposes. It is also used as a source of information for construction of other workloads. This subsystem has the capability of modifying the LRU designators on his signal list to generate a modified signal list representing a different system.

Once the workload (Signal List) data base has been satisfactorily cleaned up,the flow proceeds to the Static Subsystem as shown in Figure B-2. The Static Subsystem consists of a set of operations, groupings, or transformations which the signals go through. This is representative of:

- grouping the signals into remote terminals
- grouping the signals within the remote terminal into words
- grouping words into messages or other such requirements

For the models implemented in this effort,the above three steps were considered representative and were so modeled.

As per Figure B-2, the remote terminal assignment is the first effort of the Static Subsystem. This effort is executed as depicted in Figure B-4. The first task in this Remote Terminal Assignment Dictionary locates the different pieces of equipment (LRU's) in different areas of the aircraft (i.e., bays, etc.). Thereby each signal is in essence catalogued by location. This signal information is totaled by signal types (input or output and generic-analog, discrete, - category) for each location or bay and Remote Terminals previously assigned by remote terminal. This information is reviewed and evaluated by the user. If the remote terminal assignment is not satisfactory, the user assigns or reassigns the remote terminals through a manual update of the LRU location and Remote Terminal Assignment Dictionary. The process is reiterated until the

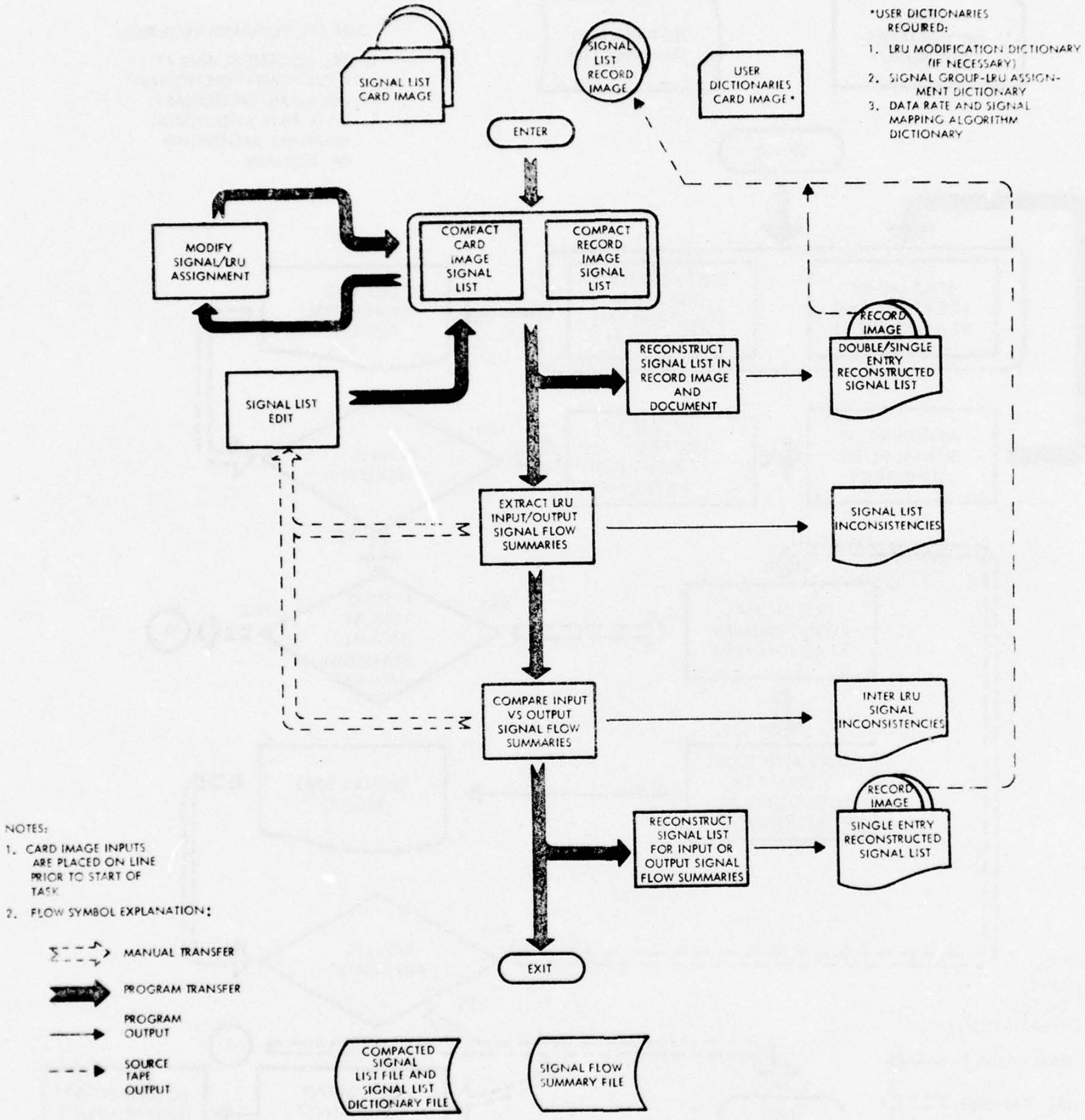

SIGNAL LIST CARD IMAGE
SIGNAL LIST RECORD IMAGE
USER DICTIONARIES CARD IMAGE *
*USER DICTIONARIES REQUIRED:
1. LRU MODIFICATION DICTIONARY (IF NECESSARY)
2. SIGNAL GROUP-LRU ASSIGNMENT DICTIONARY
3. DATA RATE AND SIGNAL MAPPING ALGORITHM DICTIONARY
ENTER
MODIFY SIGNAL/LRU ASSIGNMENT
COMPACT CARD IMAGE SIGNAL LIST
COMPACT RECORD IMAGE SIGNAL LIST
SIGNAL LIST EDIT
RECONSTRUCT SIGNAL LIST IN RECORD IMAGE AND DOCUMENT
RECORD IMAGE
DOUBLE/SINGLE ENTRY RECONSTRUCTED SIGNAL LIST
EXTRACT LRU INPUT/OUTPUT SIGNAL FLOW SUMMARIES
SIGNAL LIST INCONSISTENCIES
COMPARE INPUT VS OUTPUT SIGNAL FLOW SUMMARIES
INTER LRU SIGNAL INCONSISTENCIES
RECONSTRUCT SIGNAL LIST FOR INPUT OR OUTPUT SIGNAL FLOW SUMMARIES
RECORD IMAGE
SINGLE ENTRY RECONSTRUCTED SIGNAL LIST
EXIT
NOTES:
1. CARD IMAGE INPUTS ARE PLACED ON LINE PRIOR TO START OF TASK
2. FLOW SYMBOL EXPLANATION:
MANUAL TRANSFER
PROGRAM TRANSFER
PROGRAM OUTPUT
SOURCE TAPE OUTPUT
COMPACTED SIGNAL LIST FILE AND SIGNAL LIST DICTIONARY FILE
SIGNAL FLOW SUMMARY FILE

Figure B-3.
Utility Subsystem Data Flow Logic Diagram

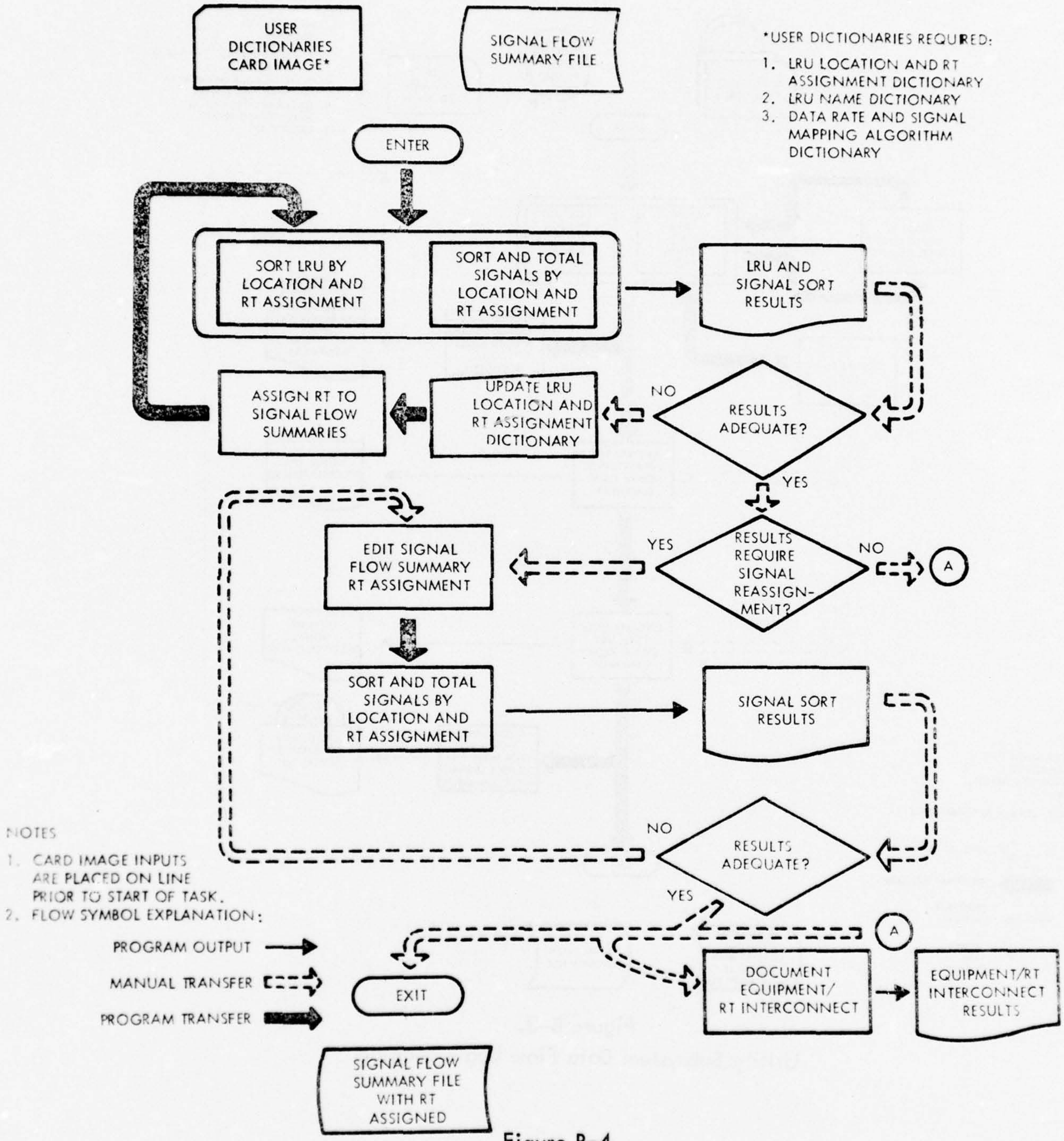

USER DICTIONARIES CARD IMAGE*
SIGNAL FLOW SUMMARY FILE
*USER DICTIONARIES REQUIRED:
1. LRU LOCATION AND RT ASSIGNMENT DICTIONARY
2. LRU NAME DICTIONARY
3. DATA RATE AND SIGNAL MAPPING ALGORITHM DICTIONARY
ENTER
SORT LRU BY LOCATION AND RT ASSIGNMENT
SORT AND TOTAL SIGNALS BY LOCATION AND RT ASSIGNMENT
LRU AND SIGNAL SORT RESULTS
NO
RESULTS ADEQUATE?
UPDATE LRU LOCATION AND RT ASSIGNMENT DICTIONARY
ASSIGN RT TO SIGNAL FLOW SUMMARIES
YES
YES
RESULTS REQUIRE SIGNAL REASSIGN-MENT?
NO
A
EDIT SIGNAL FLOW SUMMARY RT ASSIGNMENT
SORT AND TOTAL SIGNALS BY LOCATION AND RT ASSIGNMENT
SIGNAL SORT RESULTS
NO
RESULTS ADEQUATE?
YES
A
NOTES
1. CARD IMAGE INPUTS ARE PLACED ON LINE PRIOR TO START OF TASK.
2. FLOW SYMBOL EXPLANATION:
PROGRAM OUTPUT
MANUAL TRANSFER
PROGRAM TRANSFER
EXIT
DOCUMENT EQUIPMENT/ RT INTERCONNECT
EQUIPMENT/RT INTERCONNECT RESULTS
SIGNAL FLOW SUMMARY FILE WITH RT ASSIGNED

Figure B-4.
Static Subsystem RT Assign Task Data Flow Logic Diagram

user is satisfied that he has the desired assignment achievable by this method. If fine tuning of signal-to-remote terminal assignment is considered desirable then the last step in the assignment process is the use of the signal flow summary edit which allows the user to interactively move the signals from one remote terminal to another (e.g. if only a few analog signals are present in a given bay, it may prove helpful if they are all processed by the same RT). This new assignment is totaled and evaluated and the process is reiterated until the user is satisfied. At that point the user calls the documentation program and documents his configuration.

Once the remote terminal assignment is complete, the data bus message is formatted in a modular sequence. As shown in Figure B-2,the first step consists of mapping the signals to words or bytes. This is done by a model-peculiar mapping algorithm which is used by MUXSIM. The specific parameter definitions are accomplished through the use of the Data-Rate and Signal Mapping Algorithm Dictionary.

The next step consists of mapping the words or bytes to messages. Again, this is done by model-peculiar mapping algorithms assisted by user parameter definitions of the particular command and control data overhead requirements. Then a computation of average bus traffic is accomplished for the fixed telemetry format traffic.

The last step represents a three-way mode of operation. If the information transfer discipline depicted by the data base is of the fixed telemetry format variety,then the fixed format scheduler of the static subsystem is a candidate operation. It interleaves and schedules the transmission of messages among the fundamental update intervals while maintaining the required transmission periodicity for the individual message. In essence, this task consists of mapping the messages into sequence (each sequence represents a group of messages which are transmitted in a contiguous sequence on the bus). These groups are then fitted into the schedule. If the data base contains only fixed telemetry format data,then this task could conclude the operation.

If, however, the data base specifies sporadic transmission requirements for the signals,then the dynamic subsystem assists the user in the task of determining the additional time delay and/or message queueing statistics for his hardware system.

Should the data base contain only sporadic data transmission specifications,then the last step could consist of using the dynamic subsystem to obtain additional data-bus loading information.

MUXSIM provides detailed documentation for the following bus data variables (see Figure B-2):

- Corrected equipment list, signal list and signal summaries.

- The signal-to-remote terminal assignment (down to pin level).

- The signal-to-message assignment (down to signal level).
- Average bus utilization.
- Message scheduling by fundamental update interval and individual loading and utilization.
- Time delay and message queueing statistics.

The following eight static models were implemented as part of the static subsystem and are available for use as part of the signal-to-word-to-message mapping operation.

Model SA - T/T Transfer

This model is representative of direct transfer of information from one remote terminal to another remote terminal. The information is grouped into messages which are separated by the following three data variables: update rate, originating terminal,and destination terminal. In addition there is a restriction to number of data bus words that can be grouped together into one message. This quantity is a user input. The operation prior to message construction was the data bus word construction. In addition to the three message separation variables, the data word construction was kept separate by signal type (generic categories - discrete, analog, etc.) because signal conversion equipment is different for different signal types and word groupings would be separated from equipment to equipment. It should be noted that these functions are representative of hardware implementation and that the actual hardware implementation approach dictates the model.

Another factor which is considered is that any signal whose origin and destination is between the same Remote Terminals was considered to be a candidate for dedicated hardwiring and was consequently ignored by the model.

The signal-to-word mapping is done in three categories. They are:

Category A - Multisignal to Uniwords

Category B - Unisignal to Uniwords

Category C - Unisignal to Multiwords

The model specific information is inputted via the Data Rate and Signal Mapping Algorithm Dictionary.

Model SB - T/C/T Transfer

This model is representative of a system where the signal information is gathered into the Central Bus Controlling Unit,reformatted, and distributed. The big difference between Model SB and Model SA is that most data is transmitted on the bus twice. Therefore, the basic approach is to bring the signal list word into the model twice. The first time the destination remote terminal is modified to central unit representation. The second time the origin remote terminal is modified to central unit representation. In addition to the dedicated hardwire considerations of Model SA, the signals which show up with central representations for origin and destination are considered to be an internal manipulation of the central unit and are consequently deleted from the system.

Model SC - Digital T/T, Discrete T/C/T

This model is representative of direct transfer of signals which by themselves constitute one or more words (Category B and C) and the gathering and dispersing of the Central Bus Controller of the multisignal word (Category A) signal type. The message overhead words are different for the two information transfer disciplines,so the two message types are differentiated when assigning the overhead words. Each type is accordingly representative of Models SA and SB.

Model SD - Hybrid Transfer

This model is representative of a combination of the Model SA and SB disciplines where an effort has been made to minimize the data bus loading by selecting for each of the update rates the discipline which has the lower data bus rate. In essence, Model SA and SB are called, the lower data rate data transfer discipline is determined as a function of update rate,and those results are selected. All message number information is renumbered after the word and message transmission discipline structure has been selected for each update rate.

Model SE - T/T Transfer With BCIU Broadcast Reception

This model is representative of the system where certain Bus Control Interface Units (smart terminals,in essence) are capable of receiving any message going to the Central (active BCIU) Control and extracting the information they need from it. The hardware implementation is such that any signal going to any of the terminal designated BCIU is grouped together. The information transfer discipline is remote terminal-to-remote terminal,as model SA.

Model SF - T/C/T Transfer With BCIU Broadcast Reception

This model is also identical to model SE with the exception that the signal transfer discipline is representative of the central gathering and dispersing unit, as in Model SB.

Model SG - Hybrid Transfer With BCIU Broadcasting

This model is like model SB,with model SE discipline instead of model SA,and model SF discipline instead of model SB.

Model SH - T/C/T Transfer (Word Shuffling)

This model is similar to model SB. The difference in discipline is that the multisignal to word (Category A) signal types are packaged in the same fashion as they were in terminal-to-terminal discipline by model SA before they are shipped to central. This model represents a system where the bit packing is done and undone at the remote stations and the central unit merely regroups the words into messages.

The following two dynamic models were implemented as part of the Dynamic Subsystem and are available to use for the dynamic bus queueing utilization computations.

Model DA - Demand Access Transfer

This model is representative of an information transfer system which consists of a fixed format data transfer foreground plus an interrupt enabled demand access first-in, first-out background. The merits of this system are anticipated to be twofold:

- Reduced bus loading
- Reduced delay in access of the sporadic data

Some of the assumptions made to keep this model simpler, thereby enabling the simulation to cycle faster, are:

- Error and failure-free environment
- Foreground-background mode similar to hybrid analog-digital real-time operation system
- Foreground with fetch messages on a fixed telemetry format command/response basis. Background processing associated with demand data access which is made in a command/response basis.
- Interrupt system which allows the central control to initiate command/response requests for the demand access data

Some other assumptions are: one, that the foreground has no sporadic data transfer,that the computed bus load for each transfer is available from the Static Subsystem in a lumped sequence for each fundamental update interval; two, that the command response for this demand access data is initiated by central. Central therefore knows the length of data transmission for each demand access message and consequently doesn't initiate a demand message transfer which could interfere with the next foreground transmission. Only demand messages which have arrived prior to the end of foreground transmission are considered for transmission during the following background period.

Model DB - Demand Access Transfer Including Considerations of Redundancy and Fault-Handling Schemes

Basically, Model DB is a more sophisticated version of Model DA. It represents a system which consists of foreground fixed format message transmission and control plus demand access background message queueing. The demand access messages are transmitted or dispatched after completing the fixed message requirement. They are transmitted on a first-in, first-out order.

This model takes into account the impact of noise on the dual redundant data bus. In this model both data buses must be impacted by a noise event during transmission of the same message for a failure to result. Bus failures are generated in this model by using a noise event of infinite duration. Either bus can be made to fail independently. The foreground dispatching of messages is on a message-by-message basis instead of just a fixed non-varying sequence group as was done in model DA. This is necessary to evaluate the impact of noise on the message. The message is also separated into the command segment and the response segment for separate evaluation. Failures can be acknowledged by either a non-responding terminal or a failure of the controller to recognize the response. A failure is also determined by a watch-dog timer event occurring before a given message response.

Each terminal can be caused to fail by occurrence of a failure event. There are two failed modes: permanent disable or intermittent (that is,a terminal which recovers from a failure after a period of time.)

Associated with each message there is a response time which is uniformly distributed. This brings up the point that under this situation a controller cannot predict the length of time for transmitting a message. In particular, the controller checks the length of time for an average message transmission and if it is less than the time to the start of the next FUI time it schedules its occurrence. However, because of the nature of a variable response or a failed response,a feature was built to allow a delay of the next FUI fixed format message transmission until the completion of the present transmission. This element,along with variances in response, contributes jitter to the fixed format transmission schedule.

Basically all the key features of model DA are incorporated into Model DB. However, it takes longer to cycle through an equivalent simulated time than it does for Model DA, and therein lies the importance of having both models.

Failure models that have no fixed FUI requirements can be formulated from either model. This is strictly a demand message transfer system.

b. How Do I Use It?

Basically the beginner or basic user will rely on the User's Manual. The overall MUXSIM system specified and thereby implemented is an interactive set of software with optional coaching. The coaching feature was implemented to make the use of MUXSIM easy for the beginner. This interactive software is the reason for the Executive Subsystem. The coaching feature can be switched off for the more mature basic user who doesn't want to be hampered with the time delay for the coaching interchange. The veteran or advanced user will make use of System Modification Design Data Manual to create programs which are tailored to his specific requirements and install them into the MUXSIM software system.

An anticipated basic user scenario of MUXSIM is described pictorally in Figure B-5 and is explained briefly in the following three steps:

(1) For the MUXSIM user, the sofware system structure can be considered to consist of two parts:

- The MUXSIM software programs which control user interaction and the actual simulation routines.
- System Definition Libraries which consist of a family of possible system configurations the user may desire, and workloads (signal lists) which the user will model into the system; a special signal list must be manually compiled by the user first.

(2) The user, sitting at an interactive CRT or TTY console, is coached by the MUXSIM Executive Subsystem through the desired sequence of programs. As he progresses:

- He will select the signal list he wants in his system model from his workload library. The Signal List, which contains the definition of each signal, and the Equipment Location Dictionary, can be compiled either from the workload (Signal List) library by using an Equipment Complement or by entering a manually prepared signal list and/or dictionaries.

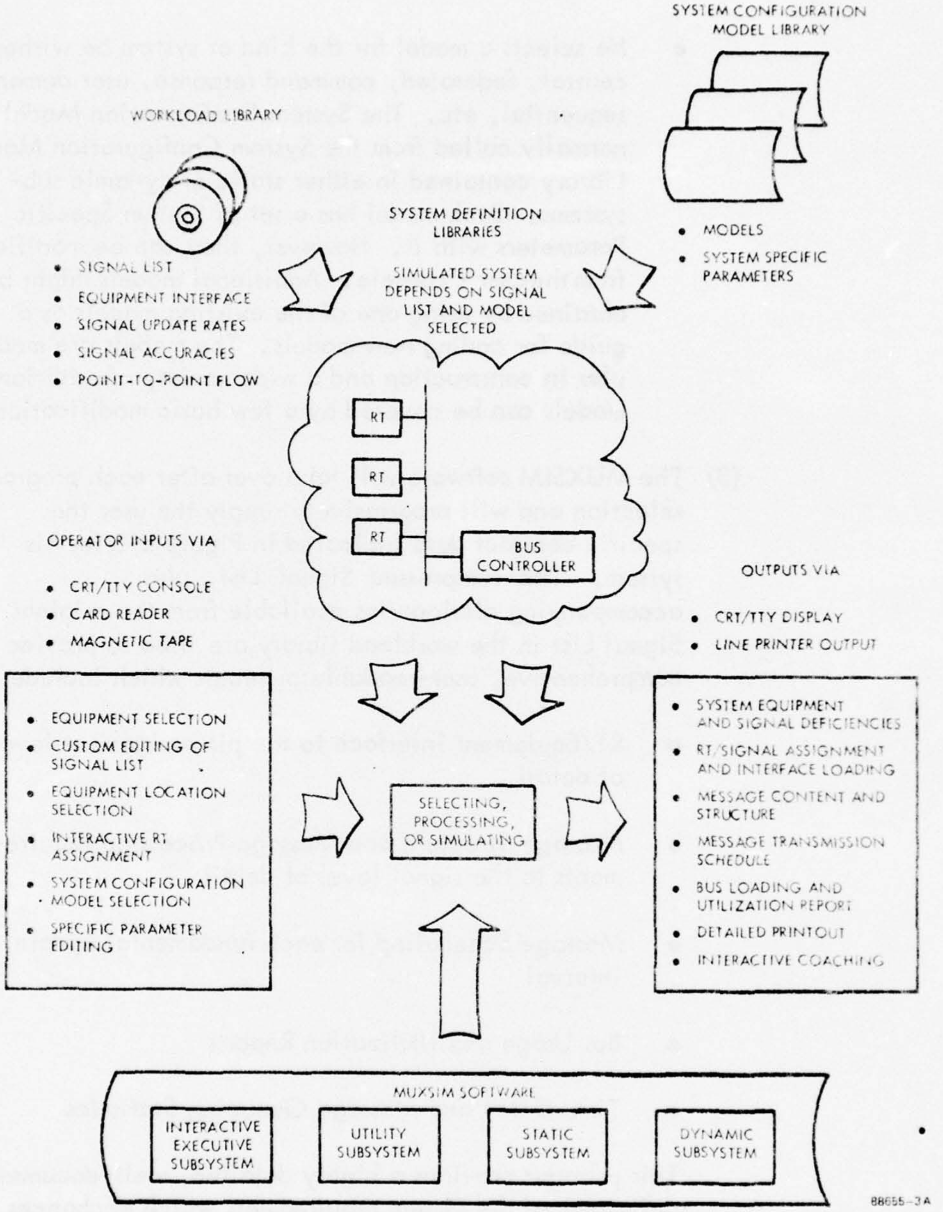

SYSTEM CONFIGURATION MODEL LIBRARY
• MODELS
• SYSTEM SPECIFIC PARAMETERS
WORKLOAD LIBRARY
SYSTEM DEFINITION LIBRARIES
• SIGNAL LIST
• EQUIPMENT INTERFACE
• SIGNAL UPDATE RATES
• SIGNAL ACCURACIES
• POINT-TO-POINT FLOW
SIMULATED SYSTEM DEPENDS ON SIGNAL LIST AND MODEL SELECTED
RT
RT
RT
BUS CONTROLLER
OPERATOR INPUTS VIA
• CRT/TTY CONSOLE
• CARD READER
• MAGNETIC TAPE
OUTPUTS VIA
• CRT/TTY DISPLAY
• LINE PRINTER OUTPUT
• EQUIPMENT SELECTION
• CUSTOM EDITING OF SIGNAL LIST
• EQUIPMENT LOCATION SELECTION
• INTERACTIVE RT ASSIGNMENT
• SYSTEM CONFIGURATION MODEL SELECTION
• SPECIFIC PARAMETER EDITING
SELECTING, PROCESSING, OR SIMULATING
• SYSTEM EQUIPMENT AND SIGNAL DEFICIENCIES
• RT/SIGNAL ASSIGNMENT AND INTERFACE LOADING
• MESSAGE CONTENT AND STRUCTURE
• MESSAGE TRANSMISSION SCHEDULE
• BUS LOADING AND UTILIZATION REPORT
• DETAILED PRINTOUT
• INTERACTIVE COACHING
MUXSIM SOFTWARE
INTERACTIVE EXECUTIVE SUBSYSTEM
UTILITY SUBSYSTEM
STATIC SUBSYSTEM
DYNAMIC SUBSYSTEM
88655-3A

Figure B-5.

Simulation Operation

- He can make any custom changes to these lists that he desires.

- He selects a model for the kind of system he wishes: central, federated, command response, user demand, sequential, etc. The System Configuration Model is normally called from the System Configuration Model Library contained in either static or dynamic subsystems. Each model has a set of System Specific Parameters with it. However, they can be modified from the user's console. Additional models might be obtained by using one of the existing models as a guide for coding new models. The models are modular in construction and a wide variety of additional models can be covered by a few basic modifications.

(3) The MUXSIM software will take over after each program selection and will progressively supply the user the specific contract data indicated in Figure B-5 for his system. The compressed Signal List, plus accompanying dictionaries available from the original Signal List in the workload library,are used to provide comprehensive, user-readable printouts which include:

- RT/Equipment interface to the pin assignment level of detail

- Message Structure and Message Processing requirements to the signal level of detail

- Message Scheduling for each fundamental update interval

- Bus Usage and Utilization Reports

- Time Delay and Message Queueing Statistics

This printout provides a highly detailed, well-documented definition of the system requirements which enchances the writing of system and module specifications.

3. LOGICAL CONSTRUCTIONS

The MUXSIM design phase identified the requirement for four major software subsystems: Executive, Utility, Static, and Dynamic, as shown in Figure B-1.

However, the major partitioning of the software construction effort was twofold:

- User Ease Software, which consists primarily of the Executive Module with its interactive features, and is designed specifically to complement the TOPS-10 operating system for user/system interface.
- Operation Software, which consists of the Utility, Static, and Dynamic modules and contains the software which does the data manipulation and algorithms necessary to perform the simulations.

The former consists of a software system which supports the user/MUXSIM interface. The primary interface is achieved between the user and the Executive module software. Other interface requirements make use of the TOPS-10 interactive features.

The latter consists of the programs which, when run batch, would produce the desired simulation results.

The two were developed separately. The requirement for the software system design effort was to evolve a set of operational software programs each of which would run in batch mode initially. Due to system functional requirements when the programs were integrated into the system, they run in pseudo-batch fashion. That is, the program would be called to run in a batch-like sequence, the main difference being that they are called interactively from the Executive subsystem.

The user ease software consists of a set of software programs which would marry the operation software to TOPS-10. The set consists of a main executive program which is a tree structure and permits accessing any operation program by following the proper calling branch. This structure is mandated by the requirement for coaching the user through a set of questions about his requirement, to direct him to the proper program. Other user-ease programs were developed to enable him to use the system efficiently. The tree structure, or main executive subsystem program, was developed standalone by using chain back to the Executive, which returned the user to the point of departure when an operation program was called. A special LDFILE program was also constructed to create the coaching text file.

The system was married by chaining from the proper point in the main executive program to the individual operation programs, and at the end of program execution chaining from the program back to the main executive program.

The main structure of the operation software system was that of progressively manipulating (mapping) the signal list through a series of three grouping criteria, each dependent on the prior one. This functional requirement translated into a logical requirement for a string of programs, each of which is driven by an input file and generates an output file which serves as the input file for the next. The last two files created one static model-dependent. In essence this progressive file manipulation provides the major interprogram communication link for the operation programs data manipulations. The approach was debugged early by running the programs batch in the proper sequence; in fact, the programs were developed in that sequence to minimize interprogram data-interface problems. Other satellite programs which are required to either document, summarize, and/or evaluate the file contents were also developed after the program which created that specific file.

a. Operation Software Construction

The top-level flow of the operation software is illustrated in Figure B-2. It shows the functional interplay of the present implementation of the three major operation subsystems: Utility, Static, and Dynamic. These major subsystems are divided into programs.

However, Figure B-2 shows the Static subsystem subdivided into five major functions. This was done because functionally it is easier to explain the operation system from this breakdown. The Executive subsystem calling sequence to the programs in the Static subsystem has an intermediate step between the call to the subsystem and the call to the individual program. This intermediate call subdivides the programs into three different groupings: RT ASSIGN (RT assignment), BUS USAGE (bus usage), and BUSSCHDL (bus schedule). This subdivision, or one much like it, was necessary simply because when a call to HELP was initiated at this level, the coach text display would exceed the CRT screen line limitations. Five subdivisions were considered excessive. The Utility subsystem has two intermediate steps, one for inputting the signal list and one for recovering the signal list. Each one calls two programs. The other programs are called directly. The Dynamic subsystems programs are called directly after call to the subsystem.

(1) Utility Subsystem Construction

As part of the software system design, the Utility sybsystem was divided into eight tasks, each to be handled by separate programs as shown in Figure B-3. The eight tasks are as follows:

- Inputting the signal list from card or card image tape and extraction of dictionaries from selected fields for the explicit purpose of reducing the on-line storage requirements for the signal list.

- The interactive editing of individual signals in the on-line signal list.

- Reconstruction of the full text signal list from the compacted signal list and associated dictionaries and the off-line (tape) storage in record image of this list and an associated printout of same list.

- Inputting the signal list from record image tape and extracting the dictionaries from selected fields, as with the card image inputs.

- Extraction of input and output signal flow summaries from the signal list, with the restriction that only signals derived can be used for the summaries pertaining to each LRU. (This is used to compare and account for double entries created by identical intersubsystem signal information being extracted from the different hardware subsystem documentation.)

- Comparison of the input versus output summaries for the matching pair. The documentation of those input and output signal flow summaries which do not pair with summaries in the complement list.

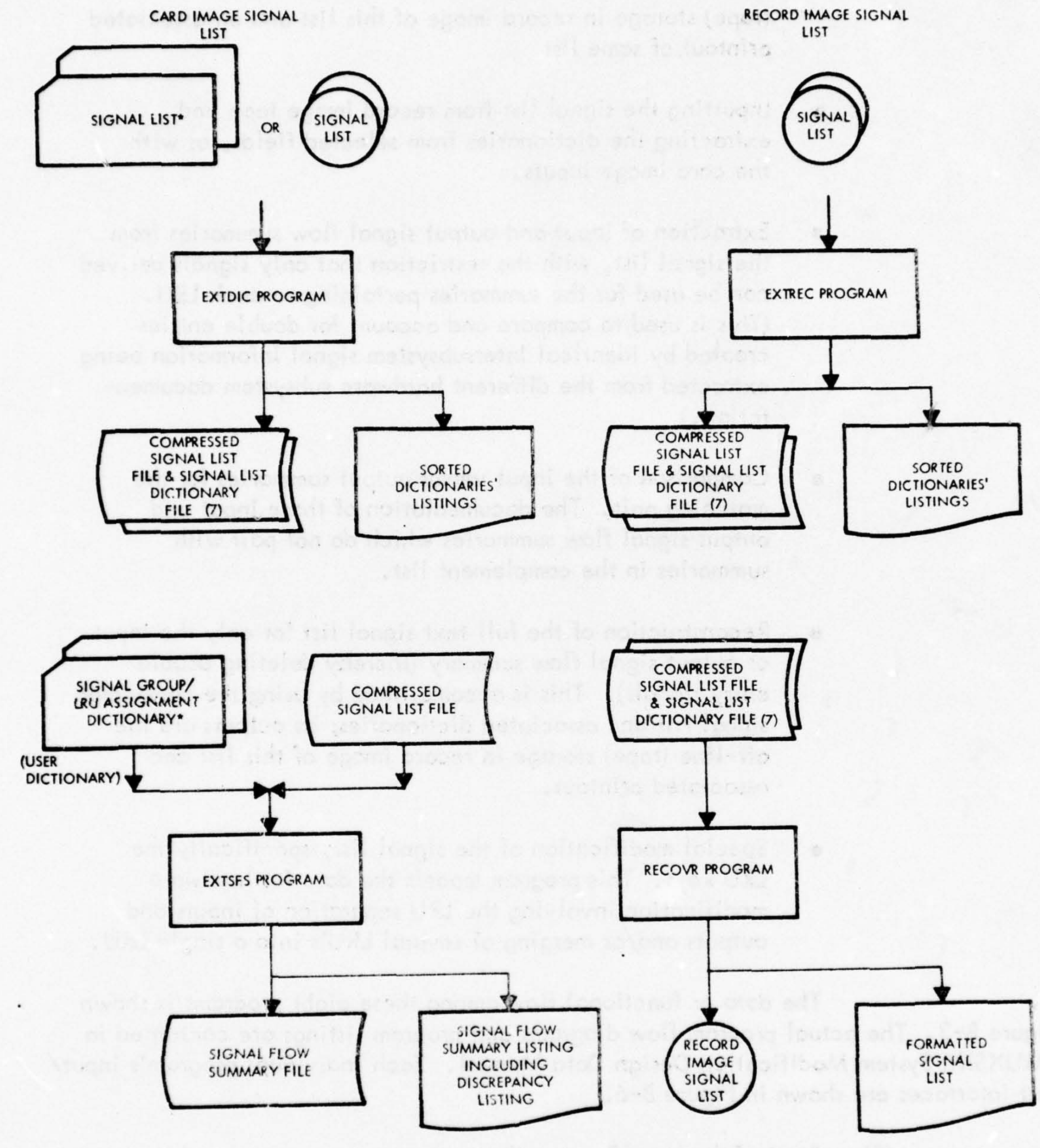
- Reconstruction of the full text signal list for only the input or output signal flow summary (thereby deleting double entry signals). This is accomplished by using the compacted signal list and associated dictionaries; its outputs are the off-line (tape) storage in record image of this list and associated printout.

- Special modification of the signal list, specifically the LRU keys. This program models the data for hardware modification involving the LRU separation of inputs and outputs and/or merging of several LRU's into a single LRU.

The data or functional flow among these eight programs is shown in Figure B-3. The actual program flow diagrams and program listings are contained in the MUXSIM System Modification Design Data Manual. Each individual program's input/output interfaces are shown in Figure B-6.

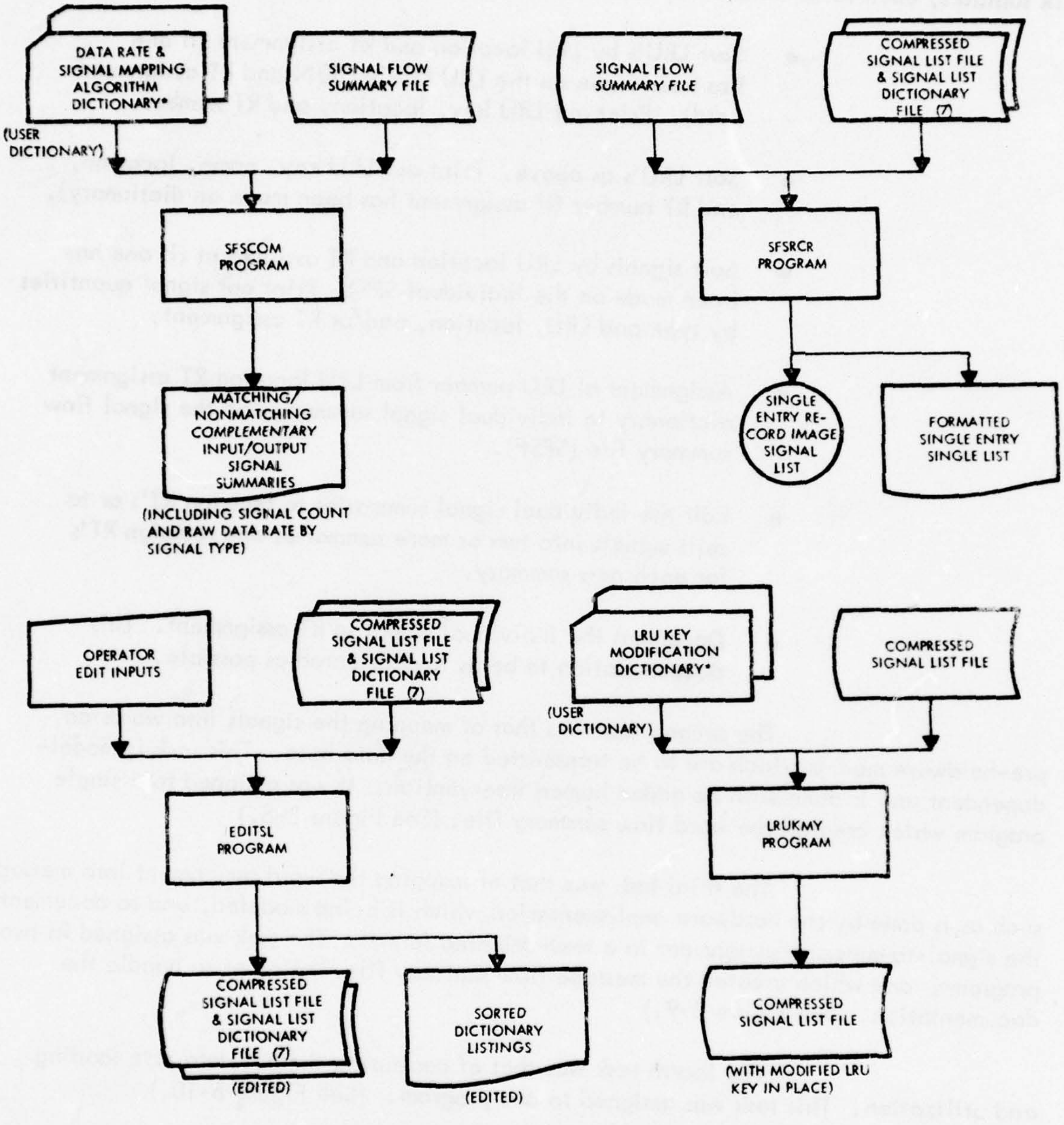
(2) Static Subsystem Construction

The Static subsystem was functionally divided into five tasks. These tasks were functionally divided into eleven programs. The first task was RT Assign; that is, the task of mapping the signals into remotely located signal collection and


CARD IMAGE SIGNAL LIST
SIGNAL LIST*
OR
SIGNAL LIST
RECORD IMAGE SIGNAL LIST
SIGNAL LIST
EXTDIC PROGRAM
EXTREC PROGRAM
COMPRESSED SIGNAL LIST FILE & SIGNAL LIST DICTIONARY FILE (7)
SORTED DICTIONARIES' LISTINGS
COMPRESSED SIGNAL LIST FILE & SIGNAL LIST DICTIONARY FILE (7)
SORTED DICTIONARIES' LISTINGS
SIGNAL GROUP/ LRU ASSIGNMENT DICTIONARY*
(USER DICTIONARY)
COMPRESSED SIGNAL LIST FILE
COMPRESSED SIGNAL LIST FILE & SIGNAL LIST DICTIONARY FILE (7)
EXTSFS PROGRAM
RECOVR PROGRAM
SIGNAL FLOW SUMMARY FILE
SIGNAL FLOW SUMMARY LISTING INCLUDING DISCREPANCY LISTING
RECORD IMAGE SIGNAL LIST
FORMATTED SIGNAL LIST

*PRIOR TO USING THIS PROGRAM OR MUXSIM FUNCTION, THE USER SHOULD CREATE A CARD IMAGE FILE WHICH IS ACCESSIBLE BY MUXSIM. THIS FILE IS USED AS INPUT BY THIS PROGRAM OR MUXSIM FUNCTION. (REFERENCE THE MUXSIM USER'S MANUAL.)

Figure B-6.
Utility Subsystem Programs and Respective Input/Output Interfaces
(Sheet 1 of 2)


DATA RATE & SIGNAL MAPPING ALGORITHM DICTIONARY*
(USER DICTIONARY)
SIGNAL FLOW SUMMARY FILE
SIGNAL FLOW SUMMARY FILE
COMPRESSED SIGNAL LIST FILE & SIGNAL LIST DICTIONARY FILE (7)
SFSCOM PROGRAM
SFSRCR PROGRAM
MATCHING/ NONMATCHING COMPLEMENTARY INPUT/OUTPUT SIGNAL SUMMARIES
(INCLUDING SIGNAL COUNT AND RAW DATA RATE BY SIGNAL TYPE)
SINGLE ENTRY RE-CORD IMAGE SIGNAL LIST
FORMATTED SINGLE ENTRY SINGLE LIST
OPERATOR EDIT INPUTS
COMPRESSED SIGNAL LIST FILE & SIGNAL LIST DICTIONARY FILE (7)
LRU KEY MODIFICATION DICTIONARY*
(USER DICTIONARY)
COMPRESSED SIGNAL LIST FILE
EDITSL PROGRAM
LRUKMY PROGRAM
COMPRESSED SIGNAL LIST FILE & SIGNAL LIST DICTIONARY FILE (7)
(EDITED)
SORTED DICTIONARY LISTINGS
(EDITED)
COMPRESSED SIGNAL LIST FILE
(WITH MODIFIED LRU KEY IN PLACE)

*PRIOR TO USING THIS PROGRAM OR MUXSIM FUNCTION, THE USER SHOULD CREATE A CARD IMAGE FILE WHICH IS ACCESSIBLE BY MUXSIM. THIS FILE IS USED AS INPUT BY THIS PROGRAM OR MUXSIM FUNCTION. (REFERENCE THE MUXSIM USER'S MANUAL.)

Figure B-6.
Utility Subsystem Programs and Respective Input/Output Interfaces
(Sheet 2 of 2)

dispersion points. This task requires a high degree of human interaction and selection. The pictorial description is shown in Figure B-7 and indicates that it was subdivided into six subtasks, each to be handled by separate programs. These subtasks are:

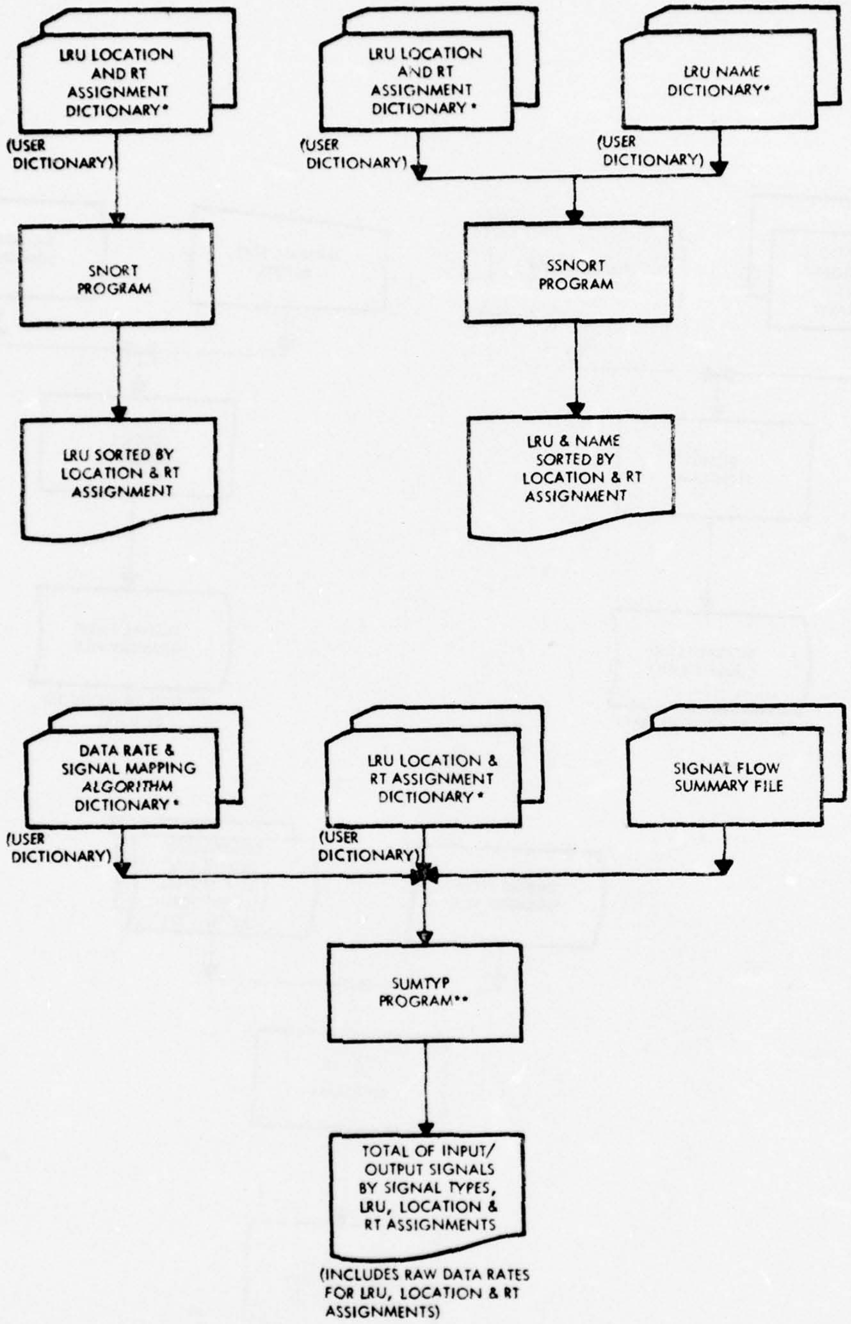
- Sort LRU's by LRU location and RT assignment (if one has been made on the LRU LOCATION and RT assignment card). Print out LRU key, location, and RT number.
- Sort LRU's as above. Print out LRU key, name, location, and RT number (if assignment has been made on dictionary).
- Sort signals by LRU location and RT assignment (if one has been made on the individual SFS). Print out signal quantities by type and LRU, location, and/or RT assignment.
- Assignment of LRU number from LRU location RT assignment dictionary to individual signal summaries on the signal flow summary file (SFSF).
- Edit the individual signal summaries to reassign RT's or to split signals into two or more summaries and reassign RT's for each new summary.
- Document the individual signal to RT assignment. This documentation to be as text-oriented as possible.

The second task was that of mapping the signals into words on pre-hardware models which are to be transmitted on the data base. This task is model-dependent and is done with no added human intervention. It was assigned to a single program which created the word flow summary file. (See Figure B-8.)

The third task was that of mapping the word assignment into messages such as is done by the hardware implementation which is being modeled, and to document the signal-to-message assignment in a text-oriented format. The task was assigned in two programs: one which created the message flow summary file; the other to handle the documentation. (See Figure B-9.)

The fourth task was that of computing the bus data rate loading and utilization. This task was assigned to one program. (See Figure B-10.)

The fifth task was that of assigning messages to fundamental update intervals (also known in telemetry as "minor frames") and computing the individual fundamental update interval's bus loading and utilization, and displaying the results in time-line display. This task was assigned to one program. (See Figure B-11.)

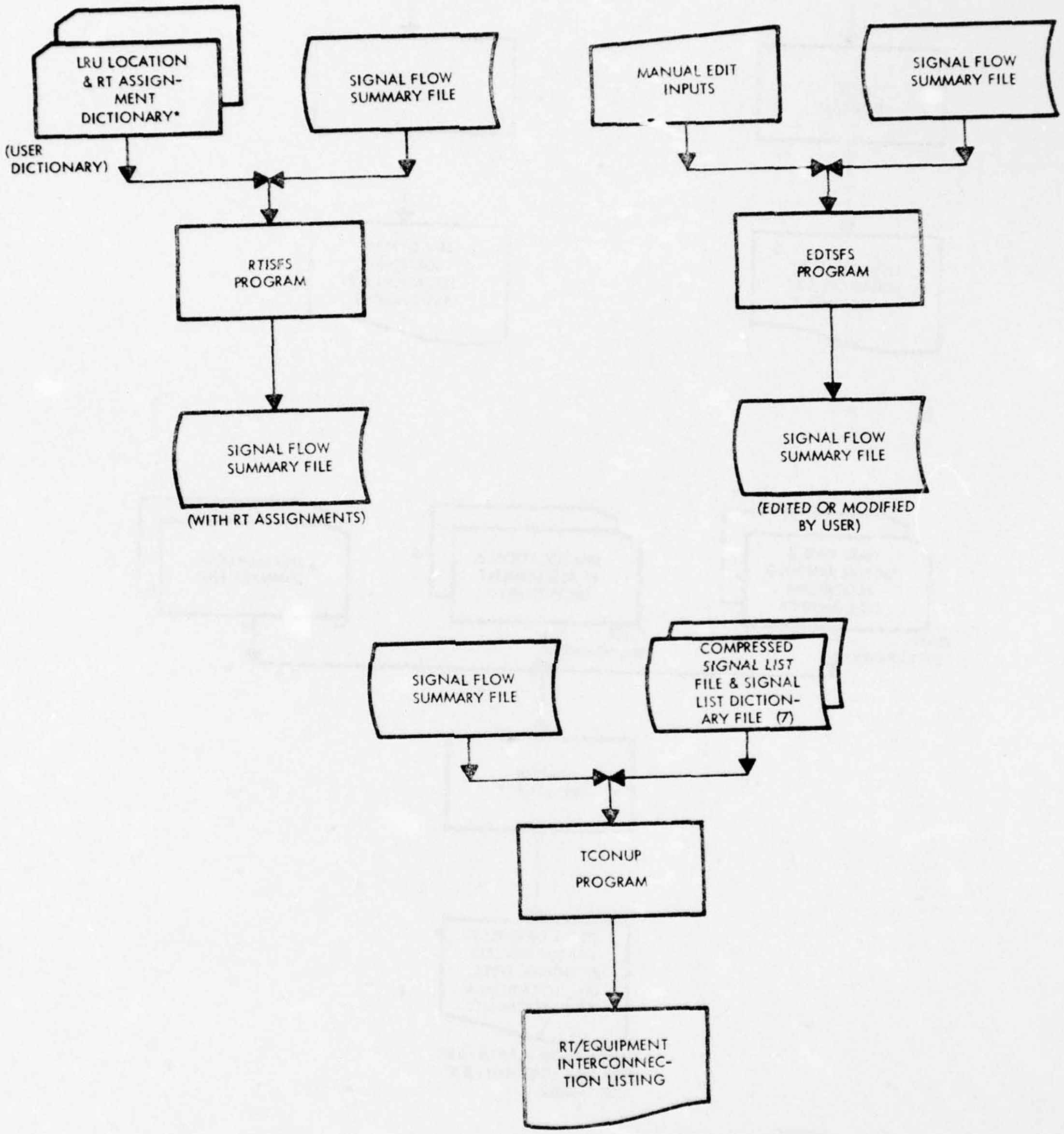

LRU LOCATION AND RT ASSIGNMENT DICTIONARY*
(USER DICTIONARY)
SNORT PROGRAM
LRU SORTED BY LOCATION & RT ASSIGNMENT
LRU LOCATION AND RT ASSIGNMENT DICTIONARY *
(USER DICTIONARY)
LRU NAME DICTIONARY*
(USER DICTIONARY)
SSNORT PROGRAM
LRU & NAME SORTED BY LOCATION & RT ASSIGNMENT
DATA RATE & SIGNAL MAPPING ALGORITHM DICTIONARY *
(USER DICTIONARY)
LRU LOCATION & RT ASSIGNMENT DICTIONARY *
(USER DICTIONARY)
SIGNAL FLOW SUMMARY FILE
SUMTYP PROGRAM**
TOTAL OF INPUT/ OUTPUT SIGNALS BY SIGNAL TYPES, LRU, LOCATION & RT ASSIGNMENTS
(INCLUDES RAW DATA RATES FOR LRU, LOCATION & RT ASSIGNMENTS)

NOTES:

*PRIOR TO USING THIS PROGRAM OR MUXSIM FUNCTION, THE USER SHOULD CREATE A CARD IMAGE FILE WHICH IS ACCESSIBLE BY MUXSIM. THIS FILE IS USED AS INPUT BY THIS PROGRAM OR MUXSIM FUNCTION. (REFERENCE THE MUXSIM USER'S MANUAL.)

** SUMTYP PROGRAM USES RT ASSIGN BY RTISES PROGRAM NOT THE DICTIONARY INPUT.

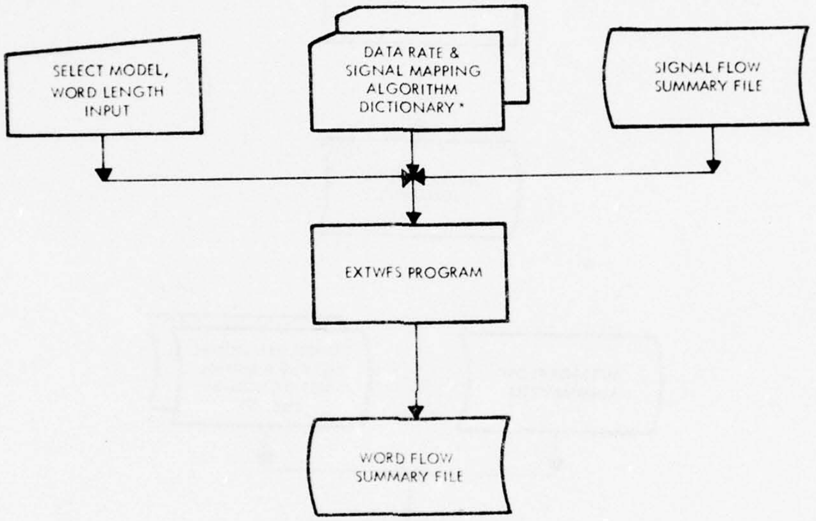
Figure B-7.
Static Subsystem RT Assign Task Programs and Respective Input/Output Interfaces
(Sheet 1 of 2)



*PRIOR TO USING THIS PROGRAM OR MUXSIM FUNCTION, THE USER SHOULD CREATE A CARD IMAGE FILE WHICH IS ACCESSIBLE BY MUXSIM. THIS FILE IS USED AS INPUT BY THIS PROGRAM OR MUXSIM FUNCTION. (REFERENCE THE MUXSIM USER'S MANUAL.)

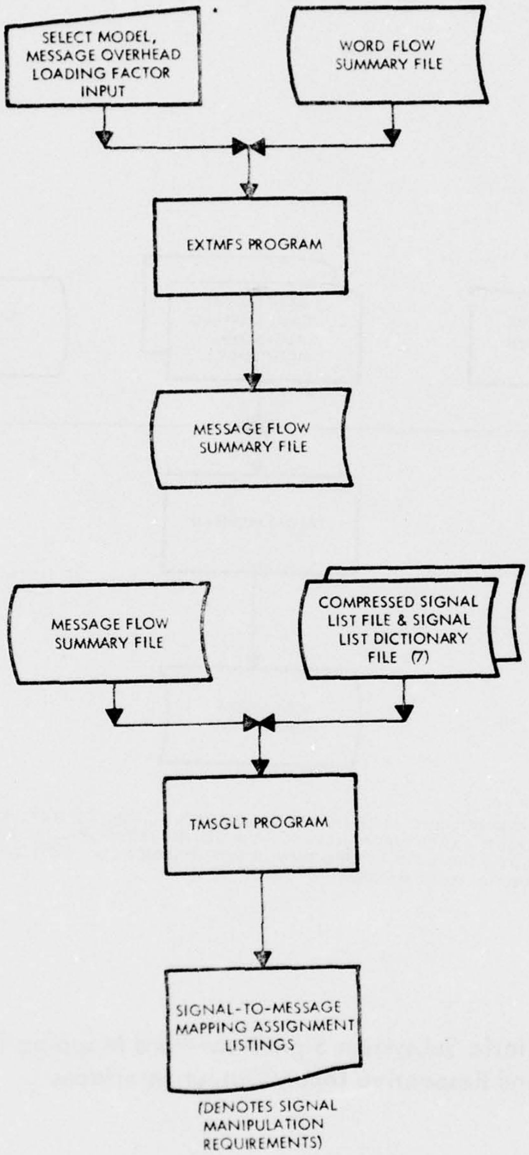
Figure B-7.

Static Subsystem RT Assignment Task Programs and Respective Input/Output Interfaces (Sheet 2 of 2)


SELECT MODEL, WORD LENGTH INPUT
DATA RATE & SIGNAL MAPPING ALGORITHM DICTIONARY *
SIGNAL FLOW SUMMARY FILE
EXTWFS PROGRAM
WORD FLOW SUMMARY FILE

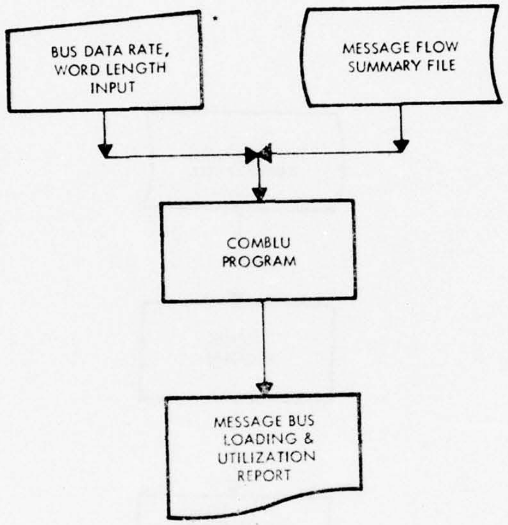
*PRIOR TO USING THIS PROGRAM OR MUXSIM FUNCTION, THE USER SHOULD CREATE A CARD IMAGE FILE WHICH IS ACCESSIBLE BY MUXSIM. THIS FILE IS USED AS INPUT BY THIS PROGRAM OR MUXSIM FUNCTION. (REFERENCE THE MUXSIM USER'S MANUAL.)

Figure B-8. Static Subsystem Signal-to-Word Mapping Task Program and Respective Input/Output Interfaces


SELECT MODEL, MESSAGE OVERHEAD LOADING FACTOR INPUT
WORD FLOW SUMMARY FILE
EXTMFS PROGRAM
MESSAGE FLOW SUMMARY FILE
MESSAGE FLOW SUMMARY FILE
COMPRESSED SIGNAL LIST FILE & SIGNAL LIST DICTIONARY FILE (7)
TMSGLT PROGRAM
SIGNAL-TO-MESSAGE MAPPING ASSIGNMENT LISTINGS
(DENOTES SIGNAL MANIPULATION REQUIREMENTS)

*PRIOR SELECTION OR INPUTS ARE CARRIED THROUGH FOR A GIVEN PROBLEM

Figure B-9
Static Subsystem Message-to-Word Mapping Task Programs and Respective Input/Output Interfaces


BUS DATA RATE, WORD LENGTH INPUT
MESSAGE FLOW SUMMARY FILE
COMBLU PROGRAM
MESSAGE BUS LOADING & UTILIZATION REPORT

*PRIOR SELECTION OR INPUTS ARE CARRIED THROUGH FOR A GIVEN PROBLEM.

Figure B-10

Static Subsystem Bus Loading and Utilization Task Program, and Respective Input/Output Interfaces

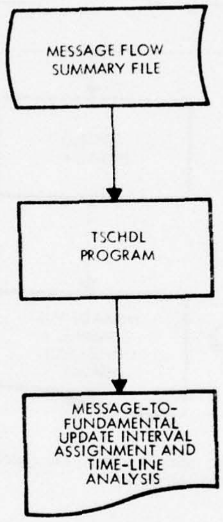

MESSAGE FLOW SUMMARY FILE
TSCHDL PROGRAM
MESSAGE-TO-FUNDAMENTAL UPDATE INTERVAL ASSIGNMENT AND TIME-LINE ANALYSIS

Figure B-11

Static Subsystem Fixed Format Schedule Task Program and Respective Input/Output Interfaces

(3) Dynamic Subsystem Construction

As part of the software system design, the Utility system was divided into two functional tasks, each to be handled by a separate program. These two tasks are:

- Simulation of a Demand Access Data Transfer System which has fixed-format data transfer via foreground dedicated transfer with a separate hardwire interrupt process to queue the demand access background messages for transmission after completion of fixed format data transfer.

- Simulation of a Demand Access Data Transfer System which has the same features as above, but takes into account impact of noise and component failures on the system, as well as system transmission jitter on the fixed-format message sequence.

The data or functional flow between the other subsystems and the dynamic subsystem is shown in Figure B-2. Each of these programs functions independently of the other. Each individual program's input/output interfaces are shown in Figure B-12.

(4) Signal List Inputs and Progressive File Construction

The basic system architecture consists of progressively manipulating the original data base from its input format into its basic message structure. There are in this MUXSIM implementation six distinct stages or evolutions in which the original workload information can exist. They consist of two input stages:

- *Card image or preliminary Signal List input*

- Record image or intermediate Signal List inputs

Both of these forms are stored off-line to MUXSIM. The second form is derived by entering and then retrieving the data from the MUXSIM input storage area. There are four on-line states in which the data base can exist, as follows:

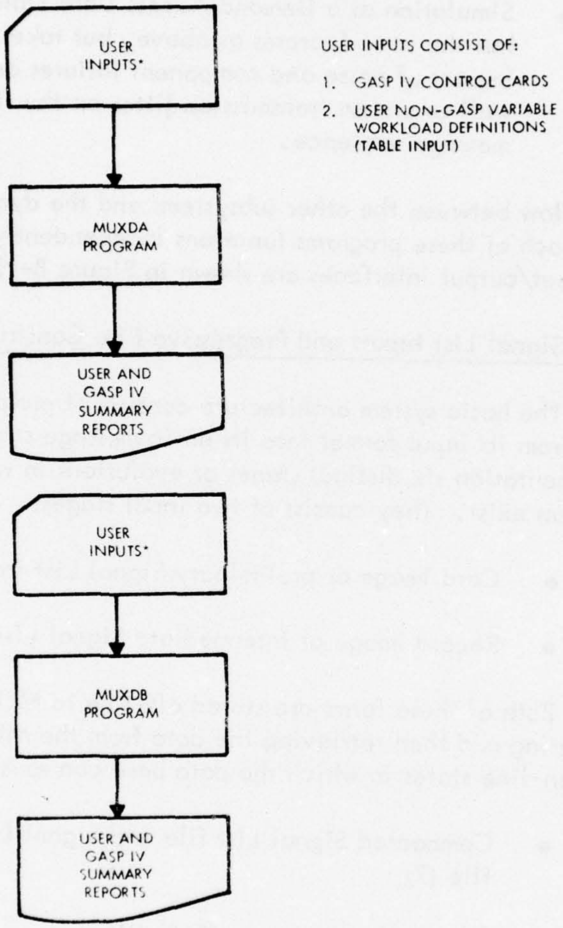
- Compacted Signal List file and Signal List Dictionary file (7)

- Signal Flow Summary (SFS) file

- Word Flow Summary (WFS) file

- Message Flow Summary (MFS) file

All other basic manipulations and computations are centered around the derivation of these progressively manipulated data inputs or files. The description of each data stage is detailed in the following paragraphs.


USER INPUTS*
USER INPUTS CONSIST OF:
1. GASP IV CONTROL CARDS
2. USER NON-GASP VARIABLE WORKLOAD DEFINITIONS (TABLE INPUT)
MUXDA PROGRAM
USER AND GASP IV SUMMARY REPORTS
USER INPUTS*
MUXDB PROGRAM
USER AND GASP IV SUMMARY REPORTS

*PRIOR TO USING THIS PROGRAM OR MUXSIM FUNCTION, THE USER SHOULD CREATE A CARD IMAGE FILE WHICH IS ACCESSIBLE BY MUXSIM. THIS FILE IS USED AS INPUT BY THIS PROGRAM OR MUXSIM FUNCTION (REFERENCE THE MUXSIM USER'S MANUAL).

Figure B-12

Dynamic Subsystem Programs and Respective Input/Output Interfaces

1. Card image or preliminary Signal List input.

   The information is contained in either cards or card image tape where the information for each signal is contained in 80-column card format spread over three card types. The data is separated by hardware subsystem within which the cards are grouped in the following sequence: all Type 1, followed by all card Type 2, and then all card Type 3. (See Figure B-13)

2. Record image or intermediate Signal List input.

   This information is contained in tape where the information for each signal is a record and each entry into the signal list is a defined field within the record. The records are sequenced by ascending signal ID number. This input is normally recreated by the MUXSIM software from input through reconstruction from the compacted signal list file and dictionary files.

3. Signal List Dictionaries and compacted Signal List files.

   These files are contained in disk or other random access mass storage areas where the information for each signal is a record and each entry into the signal list is a field. However seven of the fields are represented by numbers which are cross-references to dictionaries which have been extracted and stored in the signal list dictionary files EXTDIC program for the seven fields in order to compress the storage size requirements for the signal list.

4. Signal Flow Summary (SFS) file.

   This file is contained in disk or other random access mass storage area. The information contained is a count of the number of signals that satisfy the following criteria(along with the criteria themselves):

   - signal input or output from LRU
   - origin LRU
   - destination LRU
   - signal type (s)
   - update rate
   - quantization word/bit

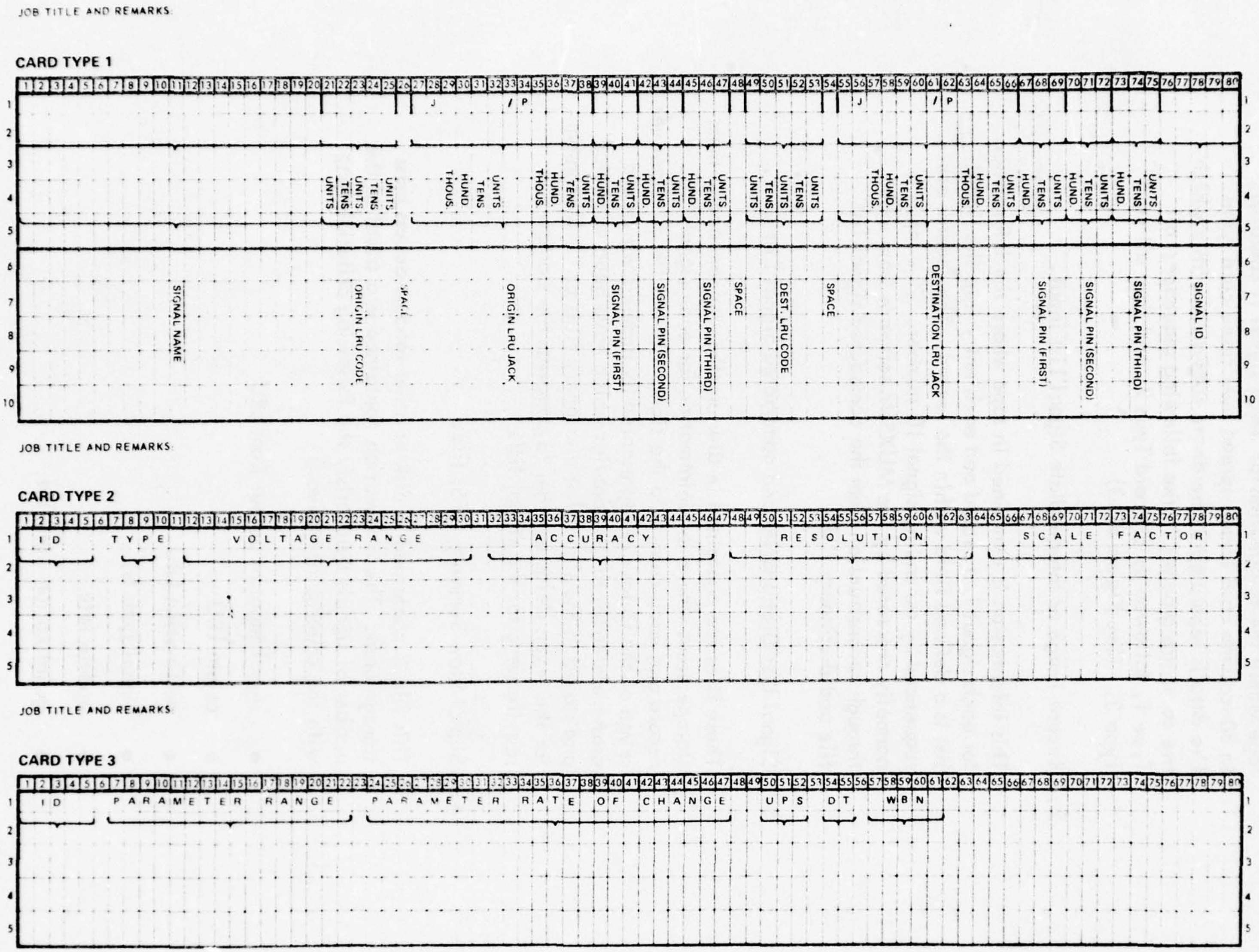


Figure B-13. 80-Column Card Creation Sheets - Types 1, 2, 3

The information consists of a record header with the variables of search and the signal ID of the signal which is part of the group. The information, for reasons of storage constraints, has a limited physical record size. It, therefore, could become necessary to carry a signal ID list logical record over multiple physical records. The source Signal List is normally extracted from documents (or other SFL's), each of which is used to describe a subsystem or equipment group. As shown or equipment group. As shown in Figure B-14, the subsystem interconnects are normally listed as outputs from one subsystem and again listed as inputs to another subsystem. The intra-system signals are normally entered into the Signal List once only. A manual accounting method for deleting these double entries is tedious and error-prone. Therefore, the problem is resolved through using the EXTSFS programs which derive the SFS for this accounting. This is accomplished by restricting the search for input and output SFS entries for each LRU to those entries in the SFL which were derived from the subsystem. This restriction is accomplished by the use of the Signal Group-LRU Assignment Dictionary, which uses the two highest digits of the Signal ID as a key for separation of the Signal List into subsystem origin. When creating the SFS, this search restriction counts the intrasubsystem signals twice, once as an output from an LRU and again as an input to another LRU. The intersubsystem signals were double listed in the Signal List and this search restriction only counts each one once.

The SFS list is separated by input and output ID designators into complementary lists. The signals which were derived from a particular subsystem portion of the list, but contain neither origin nor destination from that subsystem, are listed with an ID designator which indicates erroneous or questionable entries. Other categories of questionable entries are those signal summaries which have either an acceptable origin or destination, but the other (origin or destination) is not contained in the system LRU list. The following is a description of the SFS ID designators:

0 - Output signal summary originating from an LRU within the system.

1 - Input signal summary destined to an LRU within the system.

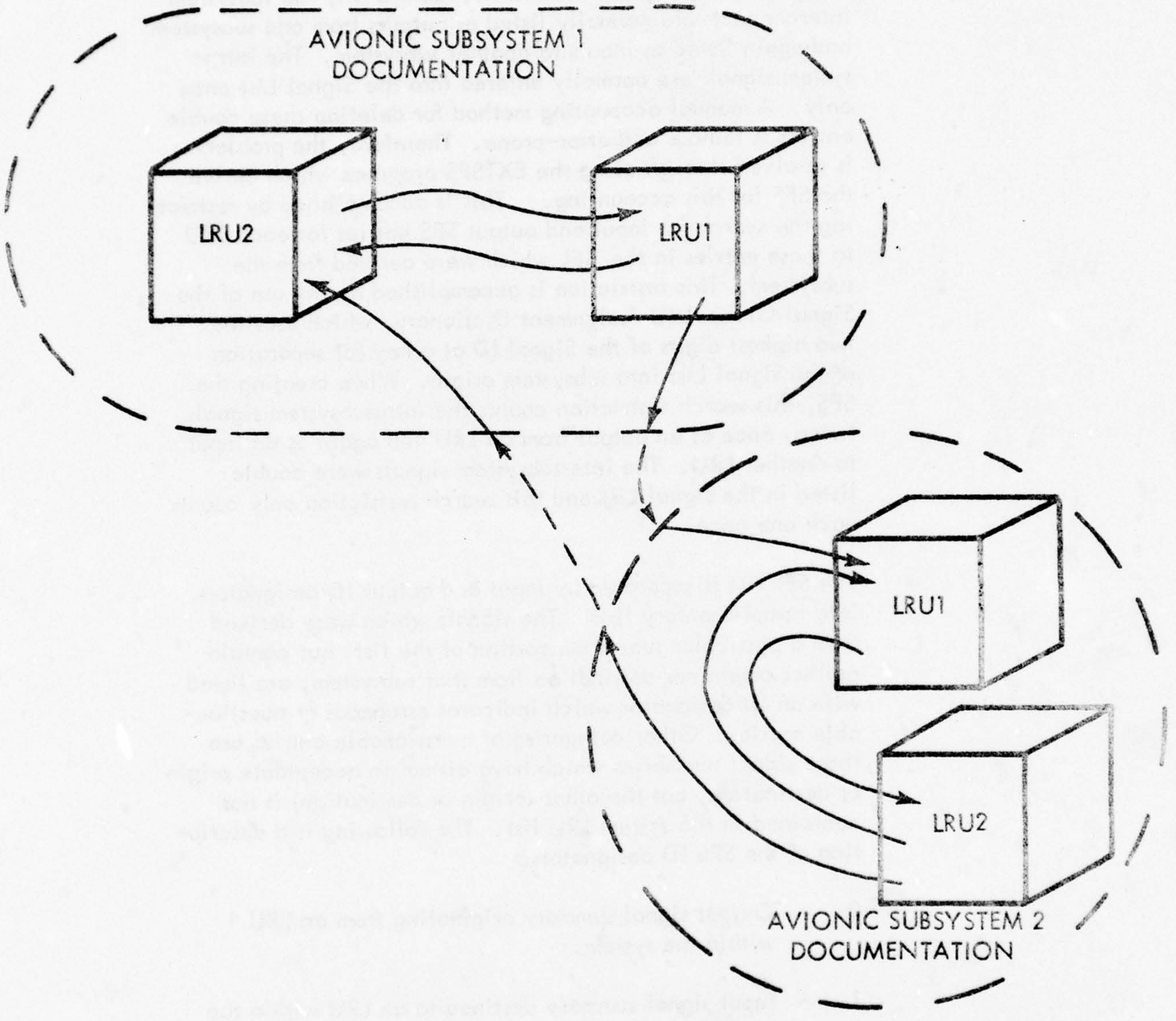


Figure B-14

Signal List Double/Single Entry for Inter/Intra-Subsystem Signal Connections

2 - Output signal summary originating from an LRU within the subsystem and destined to an LRU not listed within the system.

3 - Input signal summary destined to an LRU within the subsystem and originating from an LRU not listed within the system.

4 - A signal contained in a subsystem portion of the signal list, but containing neither an origin or destination LRU from that subsystem.

After the SFS is corrected for all SFS with ID designators 2, 3, and 4, or the user is satisfied, the input and output (1,0) are matched. All nonpaired SFS entries are listed as erroneous or nonmatching entries. When the data base is corrected so that the complementing lists match summary for summary, each half represents the signal content for the system defined by the signal list.

5. Word Flow Summary (WFS) file.

This information is usually stored in disk file. Each entry into the WFS file represents a set of signals which match the search criteria required for mapping that set of signals into words. The principal information contained in the entry is the number of words generated as a result of that mapping operation of that set of signals.

Like the SFS, each WFS logical record entry can consist of multiple physical records. The basis entry structure contains a record header which consists of the variables of search for that mapping and a list of the signal ID of those signals which form that group. The signal ID in this can also overflow into multiple records.

The information to create the WFS in this application was derived from the SFS inputs; it can, however, be derived from the compacted signal list if program modifications are made to incorporate the origin and destination RT assignment to the signal records and the signal list is purged of double entries.

The search criteria for grouping the signals is model-dependent. The search is controlled by the EXTWFS program of the static subsystem. The model selection is made interactively by the user. The search criteria for the different static models usually consist of the following variables which are contained in the individual SFS header and amended by the EXTWFS program in accordance to the specific model requirement.

- Update Rate
- Origin Remote Terminal
- Destination Remote Terminal
- Signal Type (Input/Output)
- Quantization Word/Bit Number

The Data Rate and Signal Mapping Algorithm Dictionary is used to input the system specific parameters. It provides the key for grouping signals by its signal type into the following three categories which are essential to the mapping operation:

CATEGORY A - MULTISIGNALS TO UNIWORD

CATEGORY B - UNISIGNAL TO UNIWORD

CATEGORY C - UNISIGNAL TO MULTIWORDS

In general the word mapping is a procedure of counting the number of words involved in the grouping. In Category A the word count is computed by dividing the number of signals in the group by the quantity of signals per word (dictionary entry) and rounding up to the next integer number. In Category B the word count is computed as the number of signals within the group. In Category C the word count for each signal is set equal to the quantization/word bit number or is set equal to a fixed number. The choice is made via the dictionary.

Therefore, as a result of the operation each entry of the WFS file represents a number of words on the data bus, the quantity of which is defined by this process.

if in future models it became necessary to map signals to bytes, this program, with minor modification, or one like it, could be used to do the task. The byte-to-word map could conceivably be modeled like the word-to-message map. The operation of creating bytes instead of words would represent a very similar operation (except possibly for quantity of bits).

6. Message Flow Summary (WFS) File.

This information is usually stored in disk file. Each entry into the MFS file represents a distinct data bus message. The message is composed of a set of signals which were associated with one or more WFS entries, all of which satisfied the criteria for being grouped together into a message. The principal information contained in the entry is the number of words required to carry the information and the total number of words (information plus overhead) required for message transmission.

Like the SFS and WFS, the MFS entry consists of multiple records with a header structure, plus a collection of signal ID's which can overflow into multiple records.

The information to create the MFS file has to derive from the prior data format, in this case the WFS file.

The criterion for grouping the words into messages is model-dependent and is contained in the Static subsystem. The model specific parameters required to establish a maximum number of words are a user input. This is the only additional information required to run EXTMFS besides WFS file.

If a program was set for additional mapping requirements, such as byte-to-word-to-message, sections of this mapping approach might be useful in byte-to-word mapping requirements.

This MFS file is the final evolution of the data base for the application considered in the initial implementation phase of MUXSIM. It serves as the information source for bus loading computation and bus scheduling requirements. Eventually, when a more sophisticated data base becomes available, this would serve as the input or

Workload Library for the Dynamic models. The data base in this case would have to define those signals involved in terms of stochastic processes, defining their information transfer or signalling requirements.

b. User Ease Software Construction

The user ease software construction stems mainly from the requirement to create a coached interactive system which makes the learning and use of the MUXSIM simple and convenient to the user. This user ease software is contained in the Executive subsystem and consists of the following functional requirements:

- System startup
- Tree structure for pointing to proper program or task selection
- Capability to automatically sequence through a set of programs in batch-like mode.
- Generate running (set) batch mode
- Copy on-line files to back-up tape
- Create on-line files from back-up tape
- A set of other secondary commands which add usage convenience, including a special program to construct run batch sequence
- A system to construct and/or modify the coaching text associated with the tree structure

The logical requirements which followed the preceding eight functional requirements resulted in the following five programs being defined:

- Program to initialize or start up MUXSIM.
- Program which steps and tracks the user through the tree structure, contains the secondary command functions, controls the sequencing through automatic set batch mode runs, and controls exit from set batch mode sequence.
- Program which loads the coaching text files for use by the tree structure. (This program is run in batch mode only and is not accessible through the tree structure main program).
- Program to copy on-line files to back-up tape.
- Program to create on-line files from back-up tape.

Unlike the other subsystems, there is no data flow involved in the Executive subsystem. The actual program flow diagrams and program listings are contained in the System Modification Design Data Manual. Each individual program's input/output interfaces are shown in Figure B-15.

A feature of particular interest to the MUXSIM user is the ease with which the task-calling mnemonics in the tree structure can be modified into terms which are easier for a particular set of uses at a given facility. The only requirement is to modify the proper file entry via a change of input card to this file. The task is tracked by the position of the entry in the file. When a calling mnemonic is entered, the file is searched and located, then the MUXSIM task which corresponds to the given entry location is called. The actual correlation between the interactive file mnemonics and program names and the tree structure that were used for the system test are shown in Figure B-16. The figure contains the interactive calling mnemonics and the program name immediately below it in parentheses. All commands, including the secondary commands, can have their calling mnemonics changed. However, for this implementation the secondary commands are parts of the main Executive (tree structure) programs; their names were left the same, no second names are shown in parentheses.

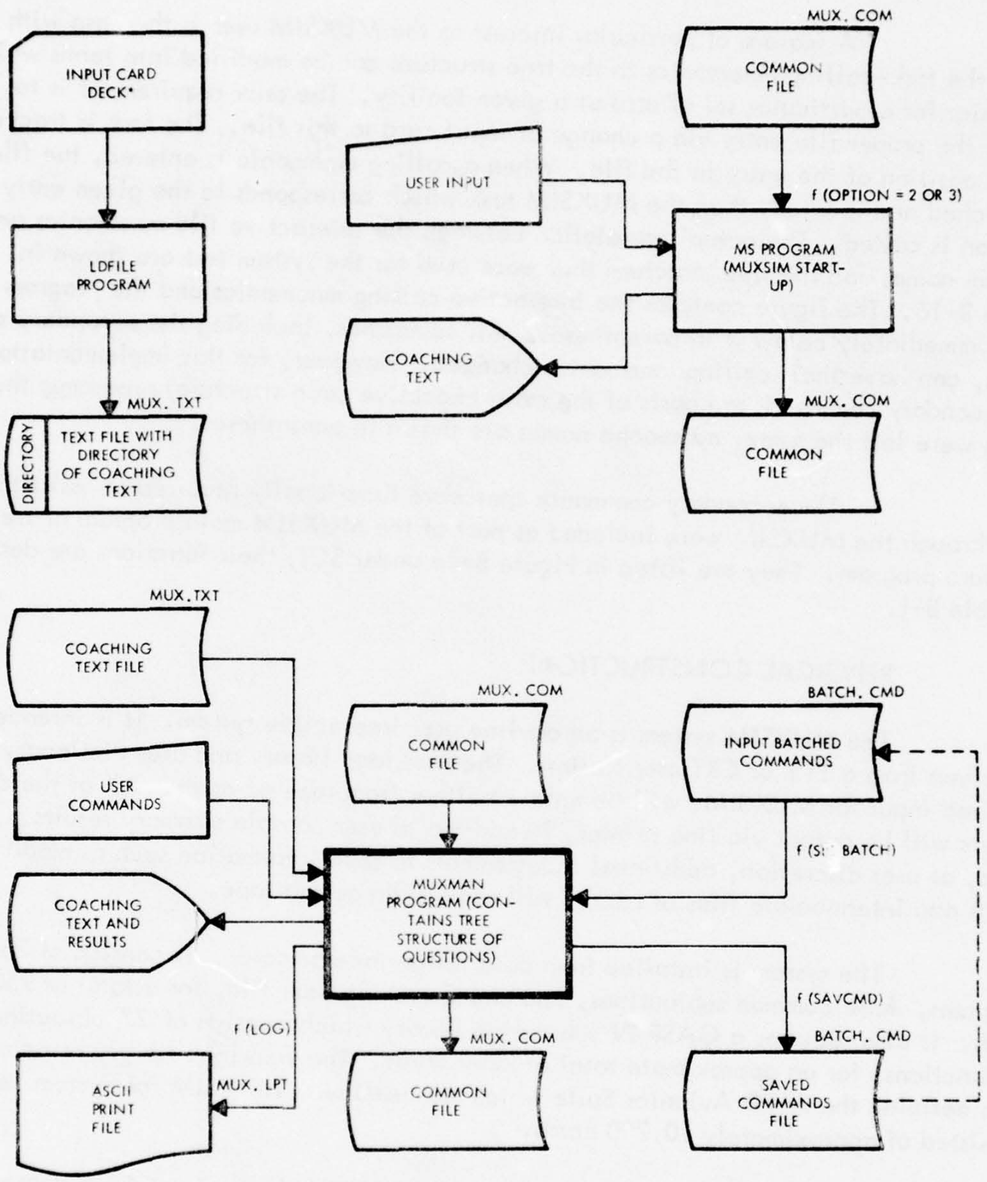
The secondary commands that were functionally required to assist the user through the MUXSIM were included as part of the MUXSIM main program or tree structure program. They are listed in Figure B-16 under SCT; their functions are described in Table B-1.

4. PHYSICAL CONSTRUCTION

The MUXSIM system is an on-line user interactive system. It is intended to be driven from a TTY or CRT user station. The workload library and user dictionary, required input for MUXSIM, will be entered either from tape or cards. All of the detailed reports will be output via line printer, in additon to user console summary results. Sometimes, at user discretion, additional requirements to save information such as modified inputs and intermediate files or results will require an output tape.

The system is installed from card image tape or cards. It consists of 26 programs, three common subroutines, and one Coaching Text file, for a total of 9500 cards. It also requires a GASP IV subroutine library which consists of 27 subroutines and 19 functions, for an approximate total of 2200 cards. The workload library associated with defining the A-7D Avionics Suite which was used with MUXSIM for System Test consisted of approximately 10,700 cards.

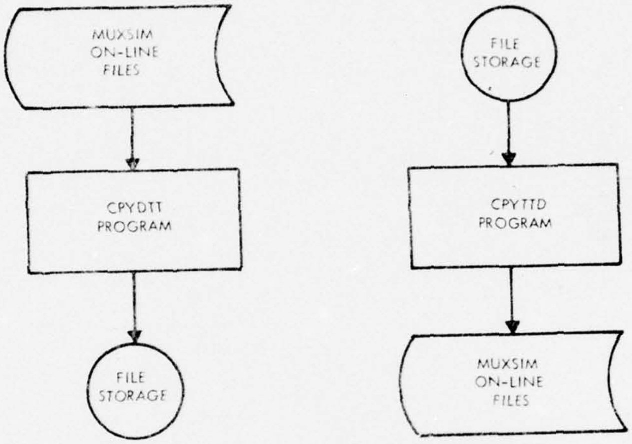
While the MUXSIM system is built in FORTRAN to enjoy a high degree of machine independence, this AFAL version has been specifically tailored to run on the host DEC System-10 with its TOPS-10 operating system. In additon, in order to install MUXSIM or re-install after modification, the system's Fortran-10 and MACRO compiler are also required.



*PRIOR TO USING THE PROGRAM OR MUXSIM FUNCTION THE USER SHOULD CREATE A CARD IMAGE FILE WHICH IS ACCESSIBLE BY MUXSIM. THIS FILE IS USED AS INPUT BY THIS PROGRAM OR MUXSIM FUNCTION. (REFERENCE THE MUXSIM USER MANUAL).

Figure B-15

Executive Subsystem Programs and Respective Input/Output Interfaces (Sheet 1 of 2)



MUXSIM ON-LINE FILES TO BE MOVED
CONSIST OF:

SGLCDS.DAT
SGLCOM.DAT
SGLDIC.DAT
SFSFIL.DAT
WFSFIL.DAT
MFSFIL.DAT
SGLADC.DIC
DRSMAP.DIC
LRUNME.DIC

LRULOC. DIC
LRUKMY.DIC
GSPDF1.DIC
GSPDF2.DIC
BATCH.CMD
SFSRTA.DAT

Figure B-15

Executive Subsystem Programs and Respective Input/Output Interfaces
(Sheet 2 of 2)

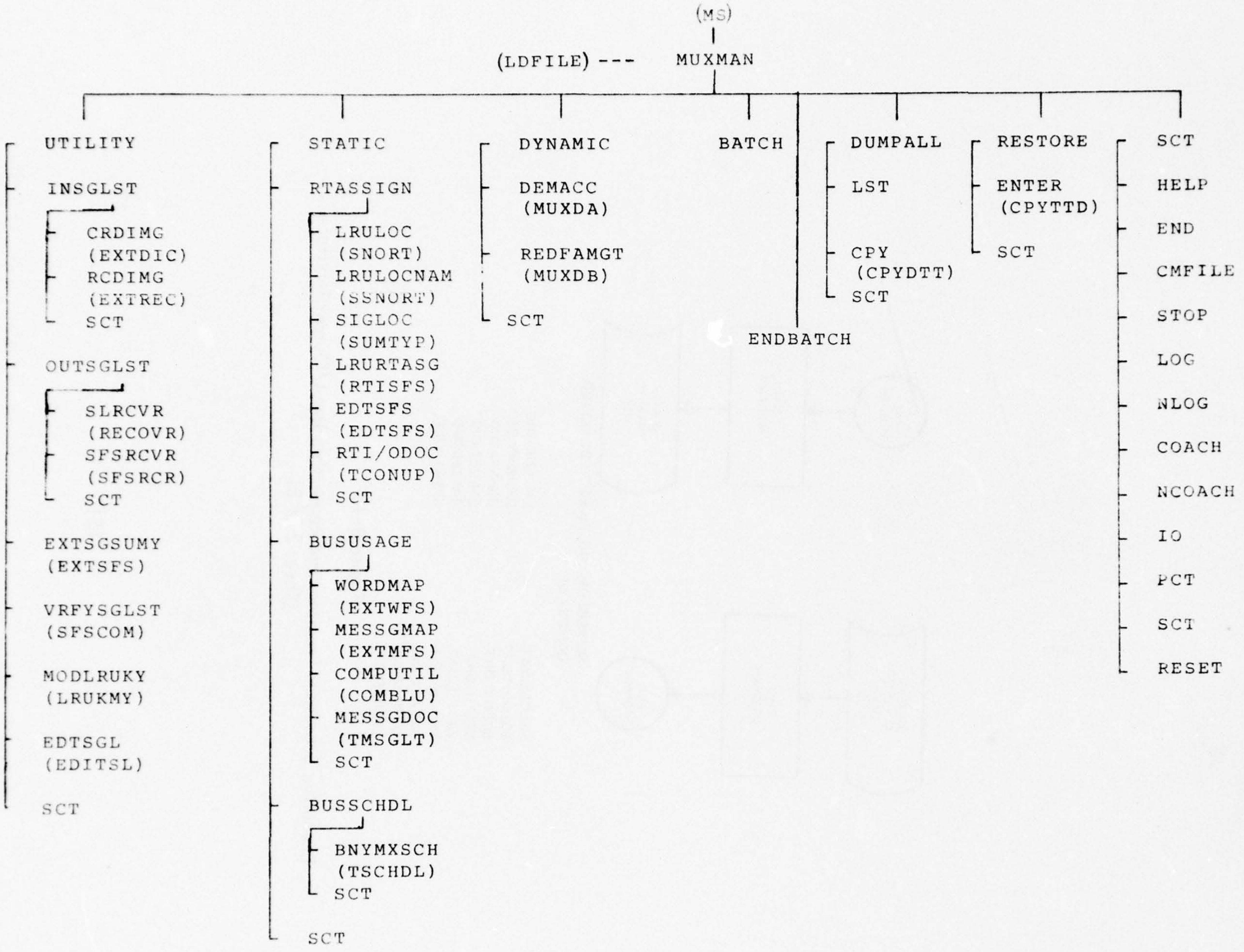


FIGURE B-16.
MUXSIM PROGRAM TREE STRUCTURE

Table B-I. SECONDARY COMMAND DESCRIPTIONS

HELP - produces the coaching text required to advise the user of his alternatives at this point in the tree structure.

END - terminates all choices at this level on the tree structure and returns the user control to the next higher level on the structure.

CMFILE - serves to create a file which is intended as the driving file when the user repeats this sequence of operations using the automatic set batch mode.

STOP - causes an exit from the MUXSIM system, and at the same time saves the location of the user in the tree structure, so that when the user re-enters MUXSIM he can choose to return to the identical location and resume his task.

LOG - copies the MUXSIM commands to an ASCII print file for record.

NLOG - halts the copying of MUXSIM commands to a print file.

COACH - is the enable command for the coach text. (It should be pointed out that at certain levels the user still has to call HELP before getting any coach text displayed.)

NCOACH - disables display of all coach text.

IO - enables the user to access MUXSIM internal file status directory for the explicit purpose of modifying the directory.

PCT - creates a display of the primary commands which are accessible at that point in the program.

SCT - creates a display of the secondary commands which are accessible at that point in the program.

RESET - resets MUXSIM files and switches to initial condition. Caution: this command recreates initial startup condition (no files defined in the internal directory).

a. Installation Requirement

The User's Manual contains a detailed section on installation procedure for MUXSIM from card image source. The procedure gives a step-by-step requirement for installation. It consists basically of:

(1) Inputting from source to the host system and creating individual source files for the 26 programs; files for the common subroutines, and a file for the coaching text.

(2) Fortran-compiling the common subroutines, with the exception that CHAIN subroutine is assembled with MACRO.

(3) Compiling the first program, linking it with the common subroutines, and saving on SAVE FILE.

(4) Proceeding with the next program until all programs are in SAVE FILE.

(5) Calling and executing LD file to install coach text.

The individual program card count, the size of disk source file in blocks, and the size of each SAVE FILE in blocks, are shown in TableB-II. These are intended to give an operator the physical requirements for system installation.

b. Usage Requirements

The Users Manual contains the details for using the MUXSIM system. The basic procedure consists of:

(1) Loading the Signal List (either from card or tape) and the User Dictionaries.

(2) Initializing MUXSIM and startup.

(3) Calling and executing the necessary MUXSIM tasks in the proper sequence to produce the required reports.

(4) Termination of task, including if considered necessary by the user, copying to tape all inputs (including modified inputs) and intermediate files for future or historical reference.

The approximate times for execution of the individual programs are shown in TableB-II. This time is referenced to the A-7D WORDLOAD Data Base. The size of the data base and the size of each of the progressive files are shown in Table B-III. The

Table B-11. DEC SYSTEM-10 MUXSIM SYSTEM PROGRAM SIZE AND TIME INFORMATION

Program Name	Card Count	Source File in Disk Blocks	Save File in Disk Block	Execution Time for A-7D Data Base Test Case (CPU Time)
MS	39	2	9	NA
LDFILE	79	3	5	0:12.7
MUXMAN	1543	72	34	NA
CPYDTT	87	11	9	NA
CPYTTD	92	11	9	NA
EXTDIC	468	55	21	13:28.4
EXTREC	323	38	16	9:13.2
RECOVR	166	20	16	2: 54.0
SFSRCR	175	21	12	1:17.4
EXTSFS	380	45	16	13:28.2
SFSCOM	393	47	16	0:22.4
LRUKMY	111	14	8	1:04.6
EDITSL	726	86	38	NA
SNORT	100	12	7	0:10.3
SSNORT	159	19	10	0:16.6
SUMTYP	551	65	25	7:52.4
RTISFS	222	27	12	0:21.1
EDTSFS	393	47	18	NA
TCONUP	247	29	13	1:04.2
EXTWFS	496	59	22	L 0:19.1 H 1:55.1
EXTMFS	451	53	21	L 0:11.4 H 0:48.6
COMBLU	330	39	19	L 0:07.1 H 0:25.8
TMSGLT	134	16	9	0:13.5
TSCHDL	575	31	21	0:11.3
MUXDA	358	21	57	1:12.9
MUXDB	887	41	67	6:38.6

Table B-III. MUXSIM INPUT PHYSICAL SIZE (A-7D WORKLOAD)

SIGNAL LIST AND PROGRESSIVE FILES

NAME	UNITS	SIZE
CARD IMAGE SIGNAL LIST	(CARD COUNT)	10,678
RECORD IMAGE SIGNAL LIST	(RECORD COUNT)	2,682
COMPRESSED SIGNAL LIST FILE SIZE	(DISK BLOCKS)	1,699
SIGNAL LIST DICTIONARY FILE SIZE	(DISK BLOCKS)	92
SIGNAL FLOW SUMMARY FILE SIZE	(DISK BLOCKS)	147
WORD FLOW SUMMARY FILE SIZE [1]	(DISK BLOCKS)	60 (MODEL DEPENDENT)
MESSAGE FLOW SUMMARY FILE SIZE [1]	(DISK BLOCKS)	44 (MODEL DEPENDENT)

USER DICTIONARIES

NAME	CARD COUNT	SOURCE FILE IN DISK BLOCKS
SIGNAL GROUP/LRU ASSIGNMENT DICTIONARY	46	7
DATE RATE AND SIGNAL MAPPING ALGORITHM DICTIONARY	19	3
LRU NAME DICTIONARY	223	30
LRU LOCATION AND RT DICTIONARY	178	24
LRU KEY MODIFICATION DICTIONARY [2]	95	13
GASP IV CONTROL CARDS AND MODEL DEFINITIONS	53/39	5/3

[1] model dependent

[2] only necessary if LRU MODIFICATION is needed to define the workload for the program.

user dictionaries sizes are also contained in Table B-III. The sizes of the various printout reports are shown in Table B-IV. These figures are intended to supply the operator a point of reference for time and size requirements for running a MUXSIM problem.

Table B-IV. LINE PRINTER MUXSIM REPORTS

PRINTOUT REPORT	LINES
SIGNAL LIST DICTIONARY (SORTED)	1,869
SIGNAL FLOW SUMMARY REPORT	1,975
SIGNAL FLOW SUMMARY RT ASSIGNED REPORT	714
FORMATTED SIGNAL LIST	16,245
MATCHING/NON-MATCHING COMPLEMENTARY INPUT/OUTPUT SIGNAL SUMMARIES	1,448
LRU SORTED BY LOCATION AND RT ASSIGNMENT	560
LRU NAME SORTED BY LOCATION AND RT ASSIGNMENT	757
TOTAL SIGNAL TYPES BY LRU LOCATION & RT ASSIGNMENT	3,348
SIGNAL-TO-TERMINAL ASSIGNMENT LIST	3,262
MESSAGE BUS LOADING AND UTILIZATION REPORT	25
SIGNAL-TO-MESSAGE ASSIGNMENT LIST	1,748
MESSAGE-TO-FUNDAMENTAL UPDATE INTERVAL ASSIGNMENT AND TIME-LINE ANALYSIS	404
DA SUMMARY REPORTS	1,470
DB SUMMARY REPORTS	1,067
MUXSIM COMMAND LOG FILE	N/A

SECTION V

VERIFICATION

1. INTRODUCTION

The verification of the MUXSIM System consisted of four parts: Module Test, System Test, Acceptance Test, and an Extended Operational Phase. The Module Test verified each program individually and was conducted after the program was built and installed on the DEC System-10. System Test was the conclusion of the initial MUXSIM implementation phase and consisted primarily of stepping through the implementation to ensure that the software package satisfied its functional, logical, and physical requirements. The Acceptance Test took the form of a final sell-off demonstration. This test requirement is contained in the MUXSIM System Test Plan (Ref. 10). Figure B-17 diagrams the MUXSIM Test Flow.

The true verification comes during the field operational phase which is to follow. Normally, the verification of simulation results is a difficult task. However, because MUXSIM is a computer-aided design and design verification tool, its usefulness is the proof of its performance, which is anticipated to be two-fold: (1) its contribution to Multiplex System Technology; and (2), more realistically, its contribution to an actual Multiplex System Design and Development Program. The simulator's true versatility depends not so much on persons being able to run the implemented models (which in this case are based on realistic implementation problems), but rather in the ability of a person to take the software system and modify it easily, then apply it to his specific problem whether it be functional or a more detailed logical model.

2. MODULE TEST

The prime objective of module test was to verify the accomplishment of milestones in the software development. The tests were required to be specific in nature in order to demonstrate the achievement of the programming goal. The tests were not required to demonstrate any specific model, as was the case in the System Test or Acceptance Test. This was in order that the model or data base development did not interfere with other software development.

a. GASP Subroutine Library Module Test

The GASP Subroutine Library is a commercial simulation package which is used as the base for the Dynamic Subsystem operation. The GASP Module Test was performed to verify that the portion of GASP used in the MUXSIM Dynamic Subsystem, the discrete event simulation feature, performed properly.

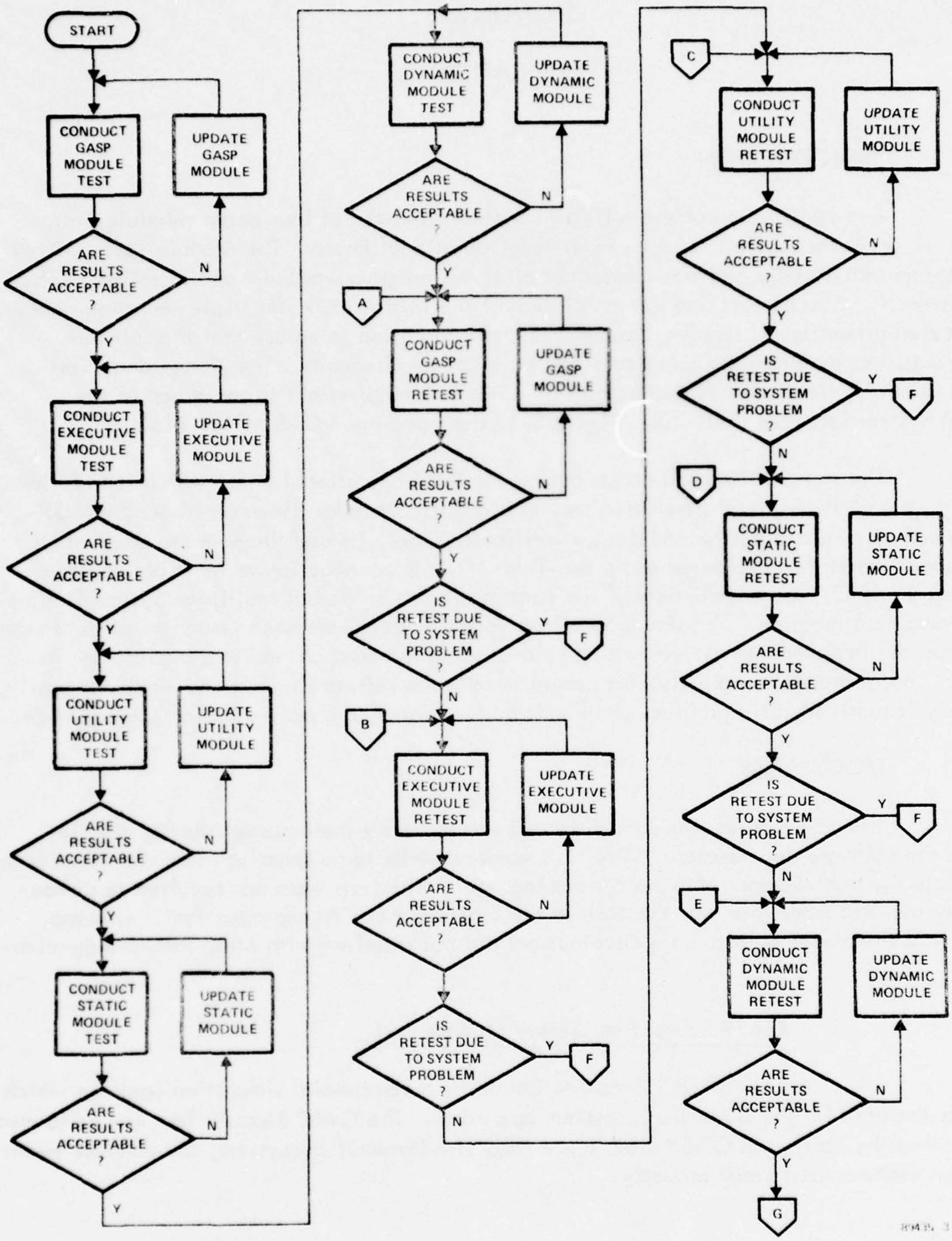


Figure B-17. MUXSIM Test Flow Diagram (Sheet 1)

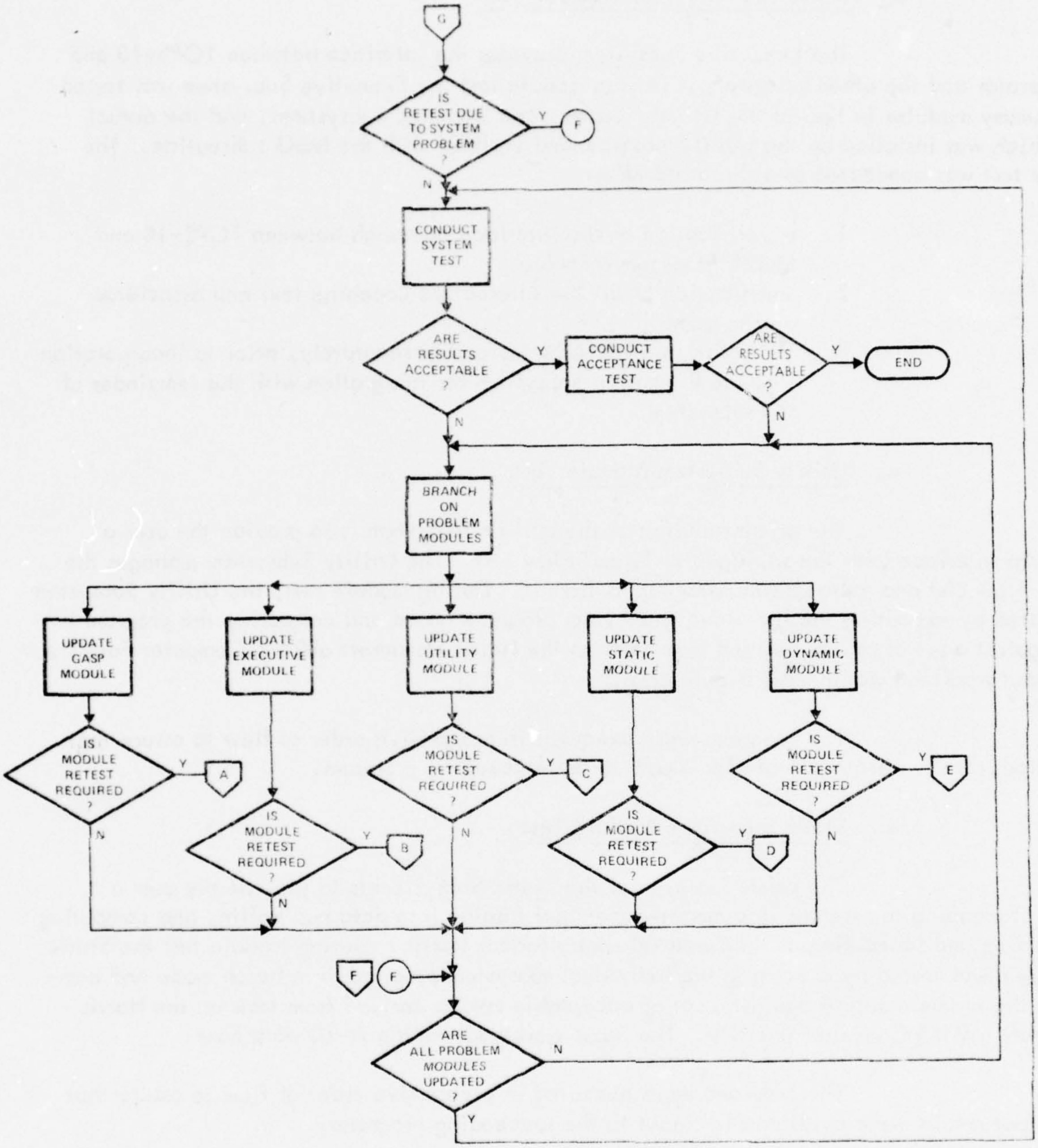

G
IS RETEST DUE TO SYSTEM PROBLEM ?
Y
F
N
CONDUCT SYSTEM TEST
ARE RESULTS ACCEPTABLE ?
Y
CONDUCT ACCEPTANCE TEST
ARE RESULTS ACCEPTABLE ?
Y
END
N
N
BRANCH ON PROBLEM MODULES
UPDATE GASP MODULE
UPDATE EXECUTIVE MODULE
UPDATE UTILITY MODULE
UPDATE STATIC MODULE
UPDATE DYNAMIC MODULE
IS MODULE RETEST REQUIRED ?
Y
A
N
IS MODULE RETEST REQUIRED ?
Y
B
N
IS MODULE RETEST REQUIRED ?
Y
C
N
IS MODULE RETEST REQUIRED ?
Y
D
IS MODULE RETEST REQUIRED ?
Y
E
N
F
F
ARE ALL PROBLEM MODULES UPDATED ?
N
Y

Figure B-17. MUXSIM Test Flow Diagram (Sheet 2)

b. Executive Subsystem Module Test

The Executive Subsystem provides the interface between TOPS-10 and the operator and the other subsystems. During module test the Executive Subsystem was tested with dummy modules in lieu of the Utility, Static, and Dynamic Subsystems, and the actual text which was installed by the LDFILE program and verified with the MSG subroutine. The module test was conducted as a threefold effort:

1. A verification of the interface approach between TOPS-10 and MUXSIM system software.
2. Verification of all the interactive coaching texr and associated option paths.
3. Verification of CHAIN subroutine separately, prior to incorporation into the Executive Subsystem for integration with the remainder of the subsystem.

c. Utility Subsystem Module Test

The prime function of the Utility Subsystem is to provide the user a system to interface with the workload or Signal Flow List. The Utility Subsystem manages the Signal Flow List and extracts simulator inputs from it. During module test, the Utility Subsystem was tested by executing the individual subsystem programs batch and comparing the program outputs against a set of results derived from tests on the Harris Datacraft 6024/5 Computer Facility. The input workload was the A-7D data base.

The programs were executed in progressive order of flow to assure that the output results were available for inputs to the succeeding programs.

d. Static Subsystem Module Test

The prime function of the Static Subsystem is to provide the user a means of mapping the signals in a model-dependent fashion into data bus traffic, and computing bus loading and scheduling of the fixed telemetry format traffic. During module test the Static Subsystem was tested by executing the individual subsystem programs in a batch mode and comparing the program output against a set of acceptable results derived from tests on the Harris Datacraft 6024/5 Computer Facility. The input workload was the A-7D data base.

The programs were executed in progressive order of flow to assure that the output results were available for input to the succeeding programs.

e. Dynamic Subsystem Module Test

The prime function of the Dynamic Subsystem is to provide the user a means of conducting simulation of random message scheduling tasks and establishing a measure of bus loading and time delay statistics. During module test the Dynamic Subsystem was tested by executing the individual subsystem programs batch and comparing the program outputs against a set of acceptable results derived from tests on the Harris Datacraft 6024/5 Computer Facility. The input workload was a synthetic workload, since an actual data base with Stochastic Signaling Definitions was not available.

3. SYSTEM TEST

The System Test demonstrated that MUXSIM has been developed with the capability to accomplish the requirements which were specified in the Statement of Work. The System Test was conducted as a two-fold exercise:

1. Module reverification
2. System model test

The first part was needed in order to verify that changes in the individual programs which occurred after undergoing module test did not impair the program's performance. The second part was the actual system test which was used to demonstrate the system's capability.

a. Module Reverification

This test was a variation of the original module test, and in some cases was expanded to include features created since the module test. The basic difference between the original test and this test is that this test served to exercise the following models:

STATIC MUX MODELS

Model Name	Information Transfer Discipline
SA	T/T Transfer
SB	T/C/T Transfer (bit shuffling)
SE	T/T Transfer with BCIU Broadcast

DYNAMIC MUX MODELS

Model Name	Description
DA	Model of MUX system using a demand-access discipline.
DB	Model of MUX system utilizing command-response information-transfer disciplines with redundancy fault-handling schemes.

b. System Model Test

This was the actual system verification test. The test consisted primarily of rerunning the individual programs tested in module reverification with the added requirement that the full interactive system, including coaching features, be on-line and was used while calling the various programs and different models.

For the final System Test, the Utility and Static Subsystems were tested using SR1 workload. The Dynamic Subsystem was tested using DSO1 and 2 workload. The acceptability criteria consisted of meeting the following three requirements:

1. The results of the System Test must compare favorably with the respective program or model test results derived during module reverification.

2. The Executive Subsystem must function correctly and every program be attainable by following the proper calling sequence.

3. The system must properly flag the user when a program is called and the required files or specific parameters are not available for use by the program.

All program results compared with the module reverification results, since both were conducted on the DEC System-10.

UTILITY AND STATIC WORKLOAD

Workload Name	Description of Workload
SR1	A-7D

Workload Name	Description of Workload
DSO1	User-Defined Workload (Model DA)
DSO2	User-Defined Workload (Model DB)

4. ACCEPTANCE TEST

This test demonstrated that MUXSIM has been developed with the capability to accomplish the requirements which were specified in the amended Statement of Work. This test consisted of a set number of exercises, experiments, or tests intended to demonstrate the various capabilities that have been designed into the system. The specific tests, objectives of each test, a general procedure, and acceptance criteria are contained in the MUXSIM System Test Plan (Ref. 10, Section 9).

The specific tests are as follows:

a. GASP IV (Test 1)

b. Load Signal List (Data Base), Extract and Verify Simulator Input, and Print the Signal List (Test 2)

c. RT Assignment (Test 3)

d. Bus Loading versus Bus Command and Control Schemes (Test 4)

e. Bus Loading versus Bus Speed (Test 5)

f. Controller Loading versus Bus Loading (Test 6)

g. Message Length Limitations versus Bus Loading (Test 7)

h. Impact of Command and Control Uncertainties on the Periodicity of the Fundamental Update Interval Starts (Test 8)

i. MUXSIM Coaching (Test 9)

j. MUXSIM Installation Test (Test 10)

k. User Demonstration

5. CONTINUAL USAGE

The operation or continual usage phase is extremely important in the reverification and usefulness of MUXSIM. It is conjectured that there will be several levels and classifications of users. They will range from the basic user to the more advanced users. The basic user is someone who is interested in learning the MUXSIM system. The rapidity of his progress will serve as a measure of the usefulness of the Coached Approach to the Executive System.

Most critical to this verification is the development of advanced users, or those who can tie MUXSIM to an actual Multiplex system design and development program, or through using MUXSIM can become contributors to Multiplex System Technology.

There are two key requirements necessary to assure continual usage:

1. Application of MUXSIM to Multiplex Design and Development Program.

2. Availability of the training necessary to encourage the development of advanced users.

A MUXSIM extension, the IDAMST Signal List, was the initial step of an effort pursuant to satisfying the first requirement. The second requirement can be covered by course work, some of which is already available. Table B-V lists the background requirements for a user's progression upward to an advanced user. From this listing are extracted the recommendations for the following course work:

- MUXSIM
- GASP
- TOPS-10 (DEC System-10 OS)

The simulator operation, conceivably, may also be supported by training in other areas, such as communication, digital, and computer science courses to broaden the user Information Transfer (Multiplex) Systems background. These courses would, of course, vary according to the user's intended hardware systems application. The three courses mentioned above are more involved with the user's skills in the mechanics of using MUXSIM.

1. MUXSIM

Training for MUXSIM may be conducted at several levels, due to the wide variations in user background. A typical family of stand-alone packages which may be offered in convenient sessions is shown in Table B-VI.

2. GASP

GASP (General Activity Simulation Program) is widely used and GASP IV training sessions are available from the developer. In addition, consulting services for both training and developmental assistance are available. Information regarding GASP training may be obtained from:

Pritskers and Associates
1201 Wiley Drive
West Lafayette, Indiana 47906
(317) 743-3287

3. TOPS-10

TOPS-10 is a well developed operating system and, as a result, various levels of training are offered. The courses are generally offered at the supplier's home office. Information regarding TOPS-10 training may be obtained from:

Digital Equipment Corporation
Software Information Services
Maynard, Massachusetts 01754

Table B-V. MUXSIM USER BACKGROUND REQUIREMENTS

Use Mode No.	Mode Description	Computer Background Requirements	Multiplex Background Requirements
1	Use existing Static & Dynamic Programs and Workloads	General understanding of MUXSIM	General understanding of multiplex systems
2	Modify existing Static & Dynamic Workloads	General understanding of MUXSIM	Knowledge of specific LRU's being changed
3	Add new MUXSIM Static Workloads	General understanding of MUXSIM	Knowledge of periodic sampling requirements for Avionics Suite Signals
4	Modify or add new Dynamic Workloads	General understanding of MUXSIM	Knowledge of dynamic signalling requirements for Avionics Suite Signals
5	Modify existing Utility or Static Programs	Knowledge of Fortran	Knowledge of the data base manipulation, multiplexing design, periodic sampling command and control techniques, and time line analysis
6	Modify Dynamic Programs	Knowledge of GASP	Knowledge of the above, plus dynamic command and control techniques
7	Add new Utility or Static Programs	Knowledge of Fortran Knowledge of MUXSIM Executive details	Extensive knowledge of the requirements for modification of same (see 5), plus knowledge of new model.
8	Add new Dynamic Programs	Knowledge of GASP Knowledge of MUXSIM Executive details	Extensive knowledge of the requirements for modification of same (see 6), plus knowledge of new model.
9	Modify MUXSIM Executive	Knowledge of Fortran Knowledge of MUXSIM Executive details Knowledge of DEC-10 Operating System (TOPS-10)	N/A
10	Change to new host computer	Knowledge of host computer architecture, operating system configuration, MUXSIM machine-dependent routines, and MUXSIM software partitioning	N/A

Table B-VI. MUXSIM USAGE AND APPLICATIONS SEMINAR

CURRICULUM

INTRODUCTION

FIRST DAY

MUXSIM Overview with emphasis on what the available documentation covers.

The MUXSIM Approach and a description of each subsystem.

BASIC USER

Utility Subsystem, its eight (8) programs and its data interface requirements.

Static Subsystem, its eleven (11) programs and its data interface requirements.

SECOND DAY

GASP IV and the Dynamic Subsystem, its two (2) programs and its data interface requirements.

Executive Subsystem, its five (5) programs and its data interface requirements. MUXSIM Installation procedure.

Demonstration of the MUXSIM coached Structure, MUXSIM Installation, and data base preparation and loading.

Hands-on execution of MUXSIM Programs. An Open Discussion of user needs and applications.

ADVANCED USER

THIRD DAY

Overview of the implementation concept for new or more detailed static and dynamic models.

Static model requirements, how to determine requirements for new static models, and modeling considerations.

Formulating techniques for evolving or creating new static models.

Utility and Executive Impact, new model impact on the Utility and Executive subsystem.

FOURTH DAY

GASP IV overview, an overview of GASP IV simulation philosophy.

GASP IV Programming requirement to simulate discrete event systems.

Dynamic model requirements, how to determine requirements for new dynamic models and modeling considerations.

Formulating techniques for evolving or creating new dynamic models.

FIFTH DAY

Demonstration of Static model modifications. Hands-on experience.

Demonstration of Dynamic model modification. Hands-on experience.

Workshop for Static-dynamic models.

Open Discussion of user special applications and general evaluation of how MUXSIM would apply.

SECTION VI

SUMMARY AND CONCLUSIONS

The MUXSIM development cycle consisted of a three-part effort: definition, design, and implementation. This Appendix covers the implementation phase. It represents 54% of the cost and 52% of the calendar time for the entire program. The prime objective during the implementation phase was to create a MUXSIM system to prove the feasibility and usefulness of such a tool.

The major problem solved by the implementation phase was that of software system design and development for the functional design of Phase II. These were three key decisions during the implementation phase: 1) passing the programs through the data and progressive file creation; 2) establishing modularity bounds for the program which are related to hardware functions, thereby facilitating model construction; and 3) the tree structure executive and handling the set of the modular programs which are defined to a set of pseudo batch programs called sequentially as needed by the user.

The use of GASP IV (1) was confirmed to be the correct decision by the effort during the program phase. GASP provides a power simulation tool constructed in an easy-to-use manner which adds versatility to MUXSIM.

Originally, the intended host system was a PDP-11/45. During the course of the development, the host system was changed to a DEC System-10. The result was a superior host system which greatly enhanced the capabilities and potentials of MUXSIM. A secondary result was that the modularization and job scope invoked by the PDP-11/45 remained, thereby assuring the success of the implementation phase.

The future success of MUXSIM depends on continual use, both to encourage its growth and increase its versatility. It is therefore necessary that an operation phase should follow, which will shake down the implementation and point to new horizons for improvements.

APPENDIX C

MUXSIM TECHNICAL NOTE BIBLIOGRAPHY

MUXSIM TECHNICAL NOTE BIBLIOGRAPHY

This appendix contains a bibliography of selected Technical Notes generated during the course of the MUXSIM program.

Technical Note Number	Title
2.2-01	DAIS Hot Bench System Definition
2.2-02	CAS DAIS LRU Configuration Definition
2.5-01	Card Coding Requirements
2.5-02	Traffic Analysis of A-7D
2.4-01	Detailed Signal Listing Column Headings
2.4-03	Detailed Signal Listing
6.0-14A	DAIS Utility and Data Management Routines
4.4-01	DAIS Multiplex Data Transfer
4.4-04	DAIS Message Sequencing
4.4-05	Message Format Modification
2.5-05	A Technique for Minimizing Data Bus Controller Message Table Pointer Editing Requirements
2.5-09	Technique for Minimizing Bus Resource Utilization on DAIS Scheduled Data Transfer
6.0-04A	DASIM1: Description, Status and Future Plans
6.0-07	DASIM6: Subroutines RDTRAF and WSORT
6.0-08	DASIM6: Subroutine WDMAP
6.0-09	DASIM6: Subroutine MSGFLO
6.0-10	DASIM7: Subroutines MREAD, MSORT, and TMELNE

Technical Note Number	Title
6.0-11	DAZIE 3
6.0-15	Time Line Analysis
2.5-06A	CAS DAIS Bus Traffic Analysis
6.0-12	Progress with GASP
5.1-02	DAIS Operability Assessment
4.4-13	ITS Redundancy Management
2.5-08	Fault Detection and Redundancy Management Models for Simulation
2.5-07	DAIS Proposed Traffic Management Scheme
3.0-01	General MUXSIM Software Organization
3.0-02	MUXSIM Executive Program
2.0-01	Technique for Using Avionics Suite Data by Nomenclature
4.0-01	PDP-11/45 RSXIID Operator Input Guide for MUXSIM at WPAFAL
3.0-03	General Utility Module Requirements
3.0-04	MUXSIM Utility/Static Module System Ops.
3.0-05	MUXSIM Program Structure
3.0-06	Utility Module Requirements
3.0-07	Utility Module Programs
3.0-08	Proposed Header Record Format for All Card/Tape/ Disc Files used in MUXSIM Effort
4.0-02	DECSYSTEM-10 User Guide for MUXSIM at WPAFAL

Technical Note Number	Note
3.0-09	Extract Word Flow Summary (EXTWFS) Program Definition
3.0-10	Extract Message Flow Summary (EXTMFS) Program Definition
3.0-11	Computation of Bus Loading and Utilization (COMBLU) Program Definition
3.0-12	MUXSIM Model DA-A Dynamic MUXSIM Model Which Uses Demand Access
3.0-13	MUXSIM Model DB-A Dynamic MUXSIM Model Which Uses Redundancy and Fault Management
3.0-14	Static Information Grouping and Handling (SIGH) Program
3.0-15	Approach for Implementing Static Models Using SIGH Program
3.0-16	Selection Criteria for SFS File Record Size
3.0-17	MUXSIM Integration Nomenclature
3.0-18	Binary Matrix Schedules (TSCHDL) Program Definition
4.0-03	Standard Program Text (MUXOPN) Definition
3.0-19	DECSYSTEM-10 Configuration
3.0-20	Program Hierarchy Nomenclature
3.0-21	Definition of TCONUP and TMSGLT
3.0-22	AFAL DEC-10 Physical Data Set Definition

REFERENCES

1. MUXSIM User's Manual, Harris ESD, June 1976.

2. MUXSIM System Modification Design Data Manual, Harris ESD, June 1976.

3. A. Alan B. Pritsker/Philip J. Kiviat, Simulation with GASP II, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1969.

4. A. Alan B. Pritsker, The GASP IV Simulation Language, Wiley-Interscience, New York, 1974.

5. Multiplex Simulator Design Study, Interim Technical Report, Harris ESD, November 1973, Vols. I and II (Phase I Report).

6. Multiplex Simulator Design Study, Second Interim Technical Report, Harris ESD, August 1974, Vols. I, II, III and IV (Phase II Report).

7. DEC System -10, Users Handbook, Digital Equipment Corp., 1975.

8. DEC System -10 Fortran -10 Language Manual, Digital Equipment Corp., 1974.

9. Data Summary for IDAMST Baseline Avionics Suite, Harris ESD, February 1976.

10. MUXSIM System Test Plan, Harris ESD, February 1976.

BIBLIOGRAPHY

SIMULATION-RELATED

J. Emshoff and R. Sisson, Design and Use of Computer Simulation Models, The Macmillan Co., New York, N.Y., 1970

G. Gordon, System Simulation, Prentice-Hall, Englewood Cliffs, N.J., 1969.

S. Greenberg, GPSS Primer, Wiley-Interscience, New York, 1972.

J. F. Hayes and D. N. Sherman, "A Study of Data Multiplexing Techniques and Delay Performance," The Bell System Technical Journal, November 1972, pp. 1983-2011.

N. Hurst and A. Pritsker, "Simulation of a Chemical Reaction Process Using GASP-IV" Simulation, September 1973, pg. 71.

H. Katzan, Jr., APL Programming and Computer Techniques, Van Nostrand Reinhold Co., New York, 1970.

P. J. Kiviat et al., The Simscript II Programming Language, Prentice-Hall, Englewood Cliffs, N.J., 1968.

L. Kleinrock, Communication Nets: Stochastic Message Flow and Delay, Dover Publications, 1964.

Francis F. Martin, Computer Modeling and Simulation, John Wiley and Sons, New York, 1968.

M. MacLaren and G. Marsaglia, "Uniform Random Number Generators," Boeing Scientific Research Laboratories, April 1964 (AD 602671).

T. M. McKinney, "A Survey of Analytical Time-Sharing Models," Computing Surveys, June 1969, pp. 105-116.

J. McLeod, Simulation: The Modeling of Ideas and Systems with Computers, McGraw-Hill Book Co., New York, 1968.

J. Mize and J. Cox, Essentials of Simulation, Prentice-Hall, Englewood Cliffs, N.J., 1968.

T. Naylor et al., Computer Simulation Techniques, John Wiley & Sons, Inc., New York, 1966.

A. M. Olson, "A Statistical Examination of Some Pseudo-Random Number Generators Having a Uniform Frequency Function," GE Apollo Support Dept., 23 August 1963.

A. Pritsker and N. Hurst, "GASP IV - A Combined Continuous-Discrete FORTRAN-based Simulation Language," Simulation, September 1973, pg. 65.

Mark Thullen, "General Elements of a Multiplex Simulation Technique," AFAL AAM.

"Description of Systems Used for Data Transmission," Subcommittee of the American Standards Committee X3, Computers and Information Processing, Communications of the ACM, October 1966, pp. 764-770.

GESIMTEL Manuals:

Volume 1 How to Model

Volume 2 Reference Guide

Volume 3 Computer In-Out

GE Corporate Education Services, Schenectady, N.Y., 1973.

M. R. Barbacci, "A Comparison of Register Transfer Languages," Department of Computer Science, Carnegie-Mellon University, March 1973.

Arla E. Weinert, "A Simscript-Fortran Case Study," Communications of the ACM, December 1967, pp. 784-792.

D. Teichnow and J. F. Lubin, "Computer Simulation-Discussion of the Technique and Comparison of Languages," Communications of the ACM, October 1966, pp. 723-741.

Ira M. Kay, "Digital Discrete Simulation Languages: A Discussion and an Inventory," Southern Simulation Services, Tampa, Florida.

Ira M. Kay, "An Over-the-Shoulder Look at Discrete Simulation Languages," Proceedings of the Spring Joint Computer Conference, 1972, pp. 791-798.

Pritsker, A. Alan B., "The GASP IV Simulation Language," Wiley-Interscience, New York, N.Y., 1974.

M. H. MacDougall, "Computer System Simulation: An Introduction," Computing Surveys, September 1970, pp. 191-209.

Y. Chu, Computer Organization and Microprogramming, Prentice-Hall, Englewood Cliffs, N. J., 1972.

C. Bell and A. Newell, Computer Structures: Readings and Examples, McGraw-Hill, New York, 1971.

SCERT descriptive literature, Comress, Rockville, Md., (301) 948-8000.

- MULTIPLEX-RELATED

Barnes, Baker and McIntosh; "The Applications of Information Transfer Techniques for Solving the Internal Communication Requirements of an Advanced Manual Bomber", AFAL-TR-72-209, Volumes I, II, and II, Radiation, Inc., 1972.

Bennett, William and Davey, James; Data Transmission, McGraw, Inc., 1965.

Blevins, J.D.; "Signal Design and Detection for Video Data Transmission", Radiation, Inc., 1971.

Borden, Perry and Mayo-Wells, Wilfrid; Telemetering Systems, Reinhold Publishing Corporation, 1959.

Butman, S., and Timor, Uzi, "Interplex - An Efficient Multichannel PSK/PM Telemetry System", IEEE Transactions on Communications, June, 1972.

Cartier, D.E.; "The Power Spectrum of PCM/FSK-AM/PM IEEE,"Transactions on Communications", July, 1973, pp 84-850.

Hamsher, Donald; Editor-in-Chief, Communication System Engineering Handbook, McGraw-Hill, Inc., 1967.

Karakash, John J.; Transmission Lines and Filter Networks, The Macmillan Company, 1950.

Kimbark, Edwards; Electrical Transmission of Power and Signals, John Wiley and Sons, Inc., 1949.

Kwan, Robert; "Advances in Digital Radio Systems," IEEE Transactions on Communications, February, 1973, pp 147-73.

Lindsey, William; "Design of Block-Coded Communication Systems", IEEE Transactions on Communication Technology, Vol. Com-15, No. 4, August, 1967, pg 525-528.

Martin, James; Telecommunications and the Computer, Prentice-Hall, Inc. 1969.

McIntosh, Brian; "Final Report for Design and Development Study for a Space Base Multiple Signal Modem (SMDS)," Vol. 1, Radiation, Inc., 1970.

Rochester, Louis; Coaxial Cables, Boston Technical Publishers, Inc., 1969.

Schnapper, M.B.; Editor, Communications, Electronics Terminology Handbook, Public Affairs Press, 1965.

Staff of White Electromagnetics, Inc., A Handbook on Electrical Filters, 1963.

Sunde, Erling; Communication Systems Engineering Theory, John Wiley and Sons, Inc., 1969.

Timor, Uzi;"Equivalence of Time-Multiplexed and Frequency-Multiplexed Signal in Digital Communications", June, 1972.

Tyree, Bennie and Bailey, James; "An Investigation of the Effects of the Error Rate of Multiple Phase-Shift Keyed Signals Through a Hard Limiter", IEEE Transactions on Communications, June, 1972, pp 36-370.

Uglow, Kenneth; "Multiplex Telemetry Modulation and Demodulation Methods", IEEE Transactions on Communication Technology, Vol. com-16, No. 1, February 1968, pp. 133-141.